Optimisation Algorithmes évolutionnaires

hepia HES-SO

Guido Bologna

Michel Vinckenbosch

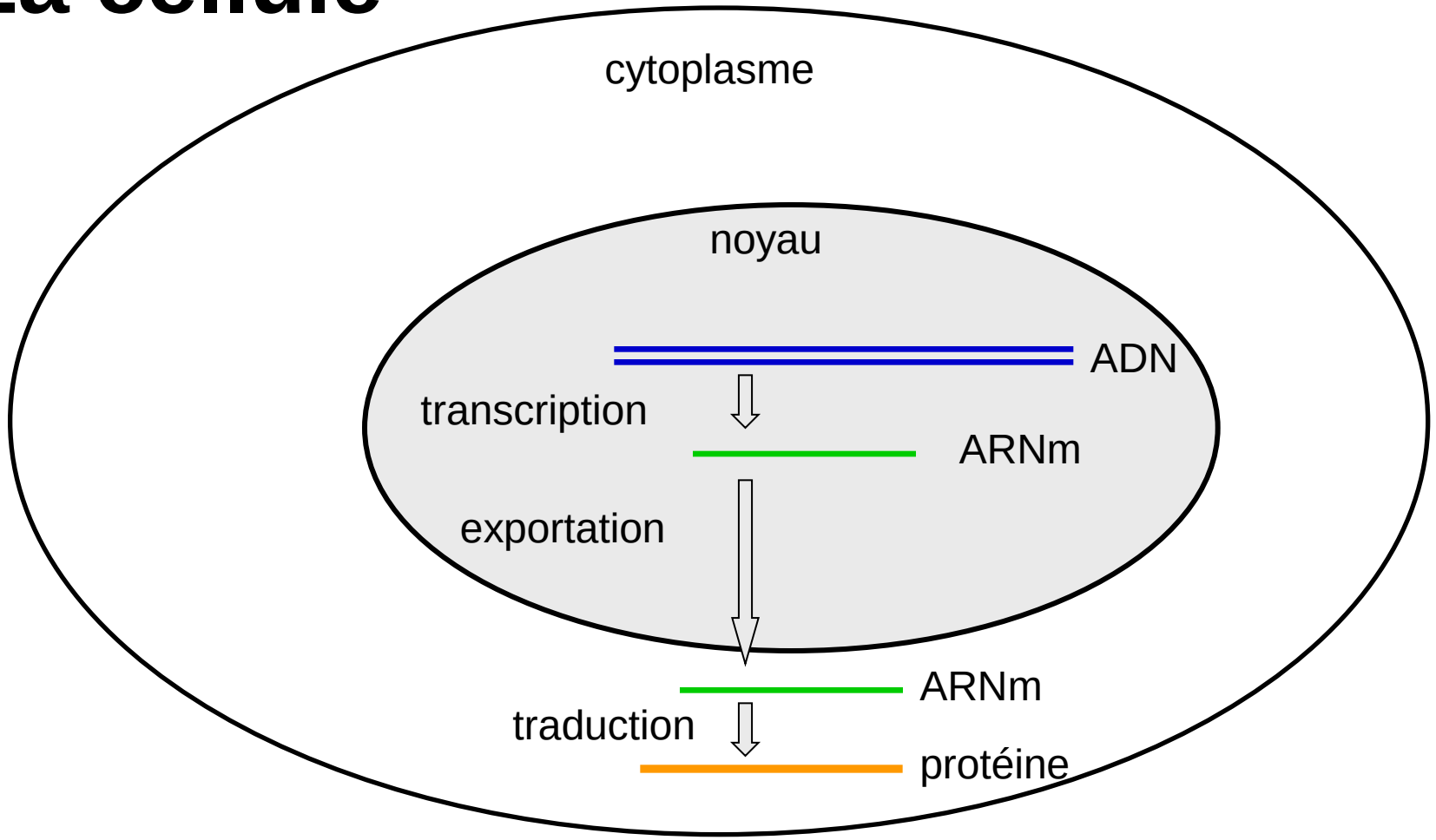
Algorithmes évolutionnaires

- ⚠ Informatique Bio-inspirée ≠ Bioinformatique
- **Fondements biologiques**
  - C. Darwin (1809-1882) A. R. Wallace (1823-1913): sélection naturelle
  - G. Mendel (1822-1884) : hérédité des caractères « génétiques »
  - T. H. Morgan (1866–1945) : théorie chromosomique
- **Algorithmes génétiques** *Genetic Algoritms*
  - GA J. Holland (1962, 1975, Ann Arbor)
- **Programmation génétique** *Genetic Programming*
  - GP J. Koza (1989, Palo Alto)
- Stratégies d'évolution *Evolution Strategies*
  - ES I. Rechenberg et H.-P. Schwefel (1965, Berlin)
  - codage réel, mutation gaussienne
- Programmation évolutionnaire *Evolutionnary Programming*
  - EP L. Fogel (1962, San Diego)
  - pas de croisement, sélection par tournoi

Bioinformatique

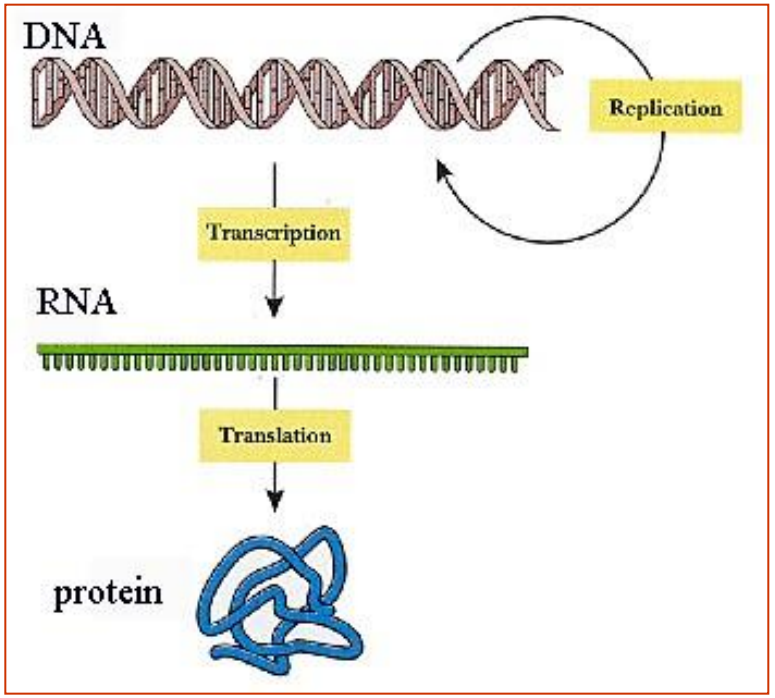
- **Analyse de séquence**
  - Séquences de génome, de transcriptome ou de protéome
  - Alignement de séquences (BLAST)
  - Séquençage et utilisation de puces à ADN
- **Modélisation moléculaire**
  - Structure 3D des protéines: conformation, sites actifs d'une enzyme, mécanismes, cibles moléculaires possibles pour cette enzyme
  - Structure 3D d'acides nucléiques (ARN et ADN)
  - Dynamique moléculaire
- **Arbres phylogénétiques**
  - Distance génétique (nombre de mutations séparant les gènes de 2 espèces)
  - Arbres: hiérarchie des espèces selon leur proximité
- **Modélisation de population**

Notions de biologie
La cellule

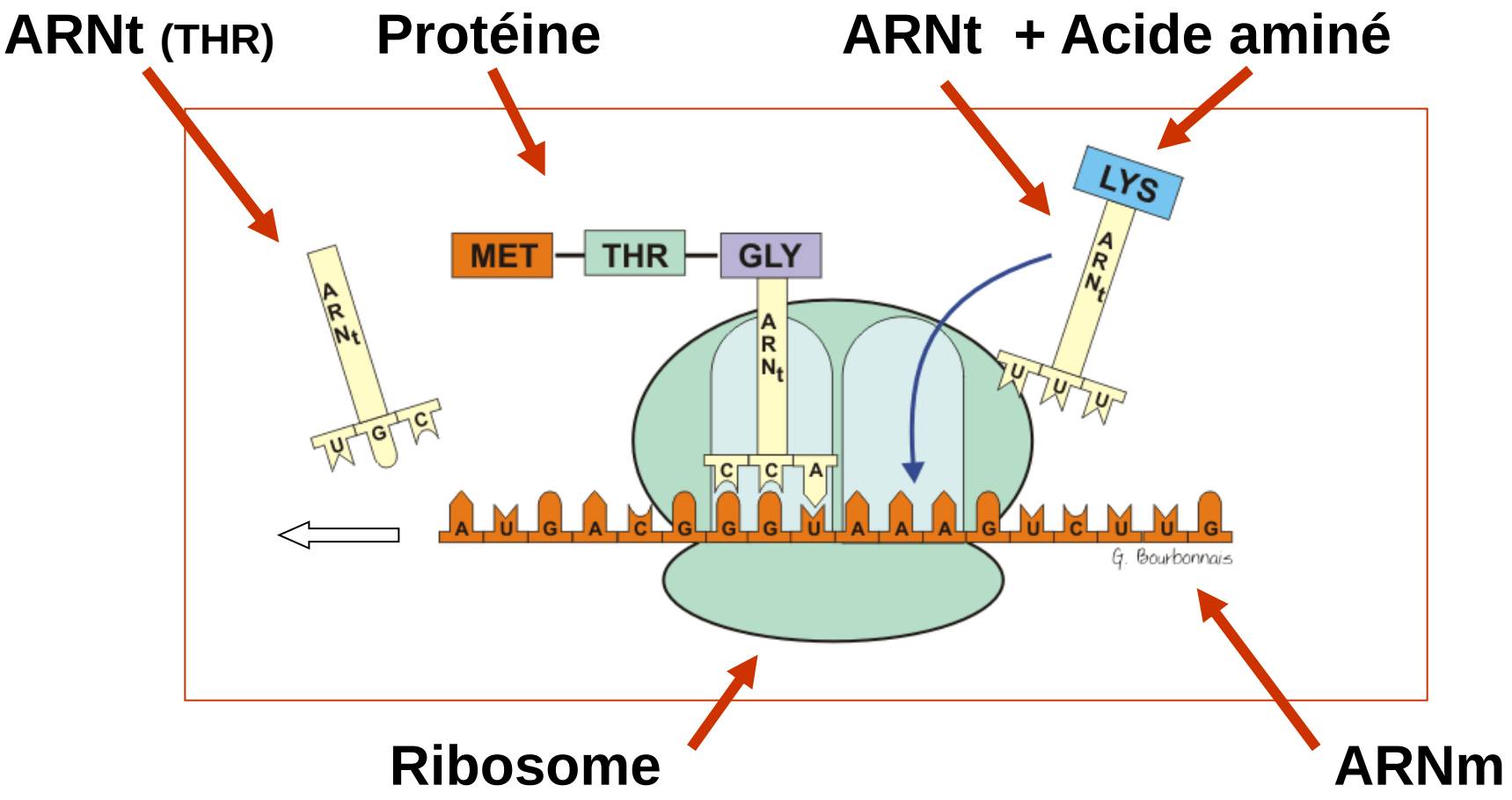


Synthèse des protéines

- Information génétique: **ADN** (**A**cide **D**ésoxiribo**N**ucléique)
- L'ADN ne sort pas du noyau. L'information passe au cytoplasme sous forme de copie **ARN** (**A**cide **R**ibo**N**ucléique)
- Synthèse des protéines dans le cytoplasme au niveau des ribosomes (traducteurs)



Mécanisme de la traduction



Codage
3 nucléotides => 1 acide aminé

- Un ARNt ne transporte pas n'importe quel acide aminé: ça dépend de l'anticodon
  - ARNt **A-A-A** transporte toujours l'acide aminé **PHE**
  - ARNt **G-A-U** transporte toujours l'acide aminé **LEU**
- Note
  - un gène peut coder la synthèse d'une protéine (ARNm)
  - un gène peut coder la synthèse d'un ARNr (ARN Ribosomal) ou d'un ARNt (ces gènes existent en des milliers de copies dans le génome)
  - gène = brin d'ADN qui est copié en ARN

Code génétique (partiel)

UUU	phénylalanine PHE	UCU	sérine SER
UUC		UCC	
UUA	leucine LEU	UCA	
UUG		UCG	
CUU	leucine LEU	CCU	proline PRO
CUC		CCC	
CUA		CCA	
CUG		CCG	

- code **dégénéré:** un acide aminé => plusieurs codons
- code **univoque:** un codon => un seul acide aminé
- code **universel** (grande majorité du vivant)
- UAA et UAG: codons **STOP**

Principe général de l'évolution darwinienne

- Le contexte de l'évolution est une **population** (d'organismes, d'objets, d'agents ...) qui survivent un temps limité puis meurent. Certains produisent une **descendance** pour les générations suivantes. Les **'mieux adaptés'** tendent à se reproduire plus
- Sur plusieurs **générations**, la composition de la population change sans qu'aucun individu ne change au cours de sa vie. Lors des générations successives, « l'espèce » évolue et s'adapte en quelque sorte aux conditions.

Principe général de l'évolution

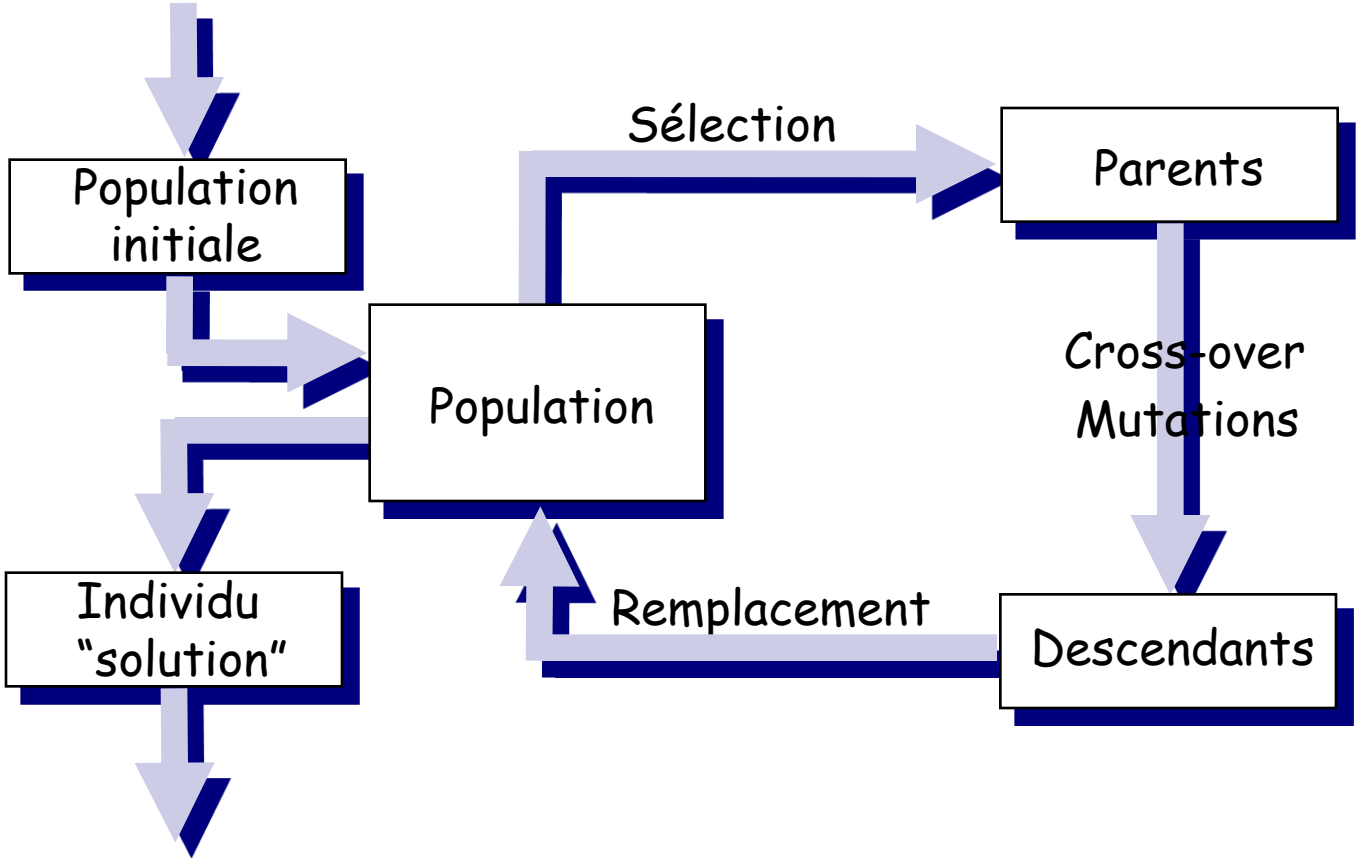
- Hérédité
  - Les descendants sont ***similaires*** aux parents
- Variabilité
  - Des changements aléatoires « perturbent » l'hérédité
  - Des recombinaisons aléatoires augmentent la diversité
- Sélection
  - Les organismes les plus adaptés à leur environnement sont susceptibles d'avoir plus de descendants

Ces trois processus sont réunis au sein d'une boucle générationnelle …

Algorithmes Génétiques: méthode d'exploration aléatoire bio-inspirée

- Sur le plan algorithmique, les AGs sont caractérisés par
  - Codage des solutions potentielles (génotypes des individus)
  - Exploration multiple (population)
  - La fonction à optimiser permet l'évaluation des individus (phénotype, performance ou *fitness*)
  - Règles de transition probabilistes (mécanismes de sélection / reproduction / mutation).
- On cherche une bonne solution à partir d'une population de solutions candidates
  - Tant qu'une bonne solution n'est pas trouvée, la population est soumise à une évolution dirigée

Principe du cycle d'évolution



Algorithme génétique

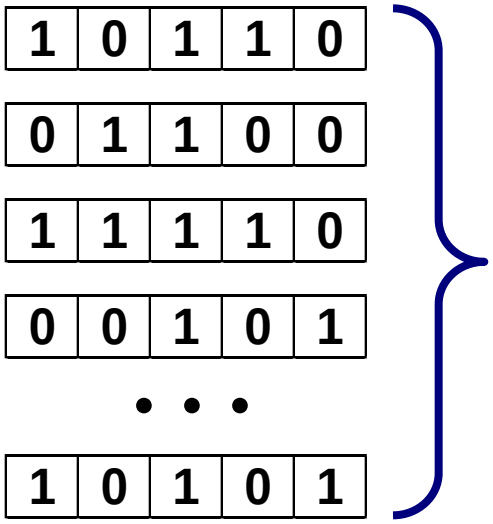
```
créer population initiale
faire
   évaluer tous les individus (avec fct fitness)

   faire
       sélectionner parents
       créer descendants par recombinaison
       faire muter descendants
   jusqu'à taille population

   remplacer tous les individus par les descendants
jusqu'à critère d'arrêt
```

Représentation et initialisation Exemple

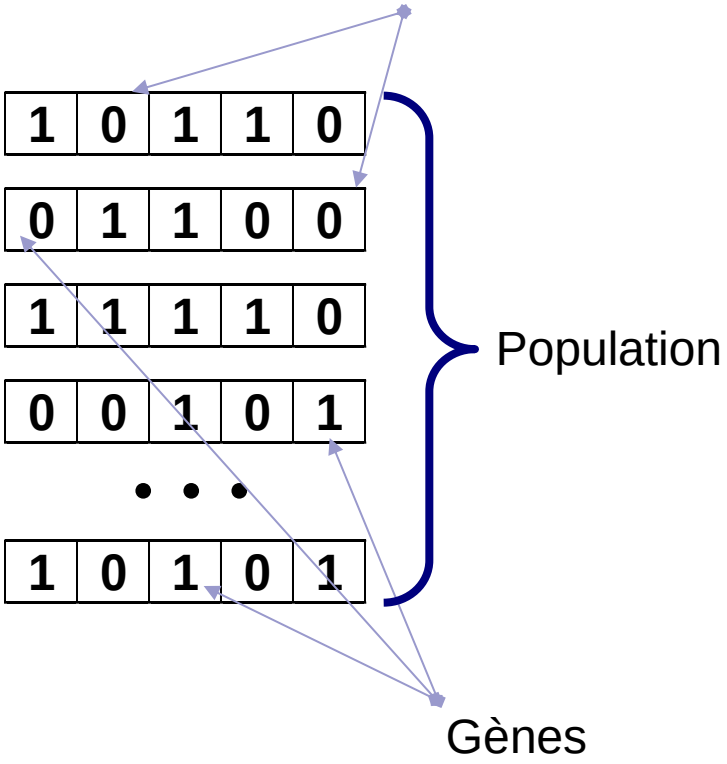
- Un individu est codé par une chaîne de bits (génotype)
- Chaque chaîne code une solution candidate
- La population initiale est aléatoire



N individus générés aléatoirement (population initiale)

Terminologie

Chromosomes, Individus ...



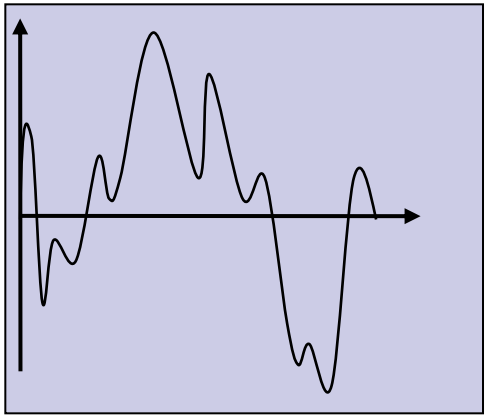
Génotype : configuration de la chaîne binaire

Phénotype : ce que le génotype représente « réellement »

0	1	1	0	0

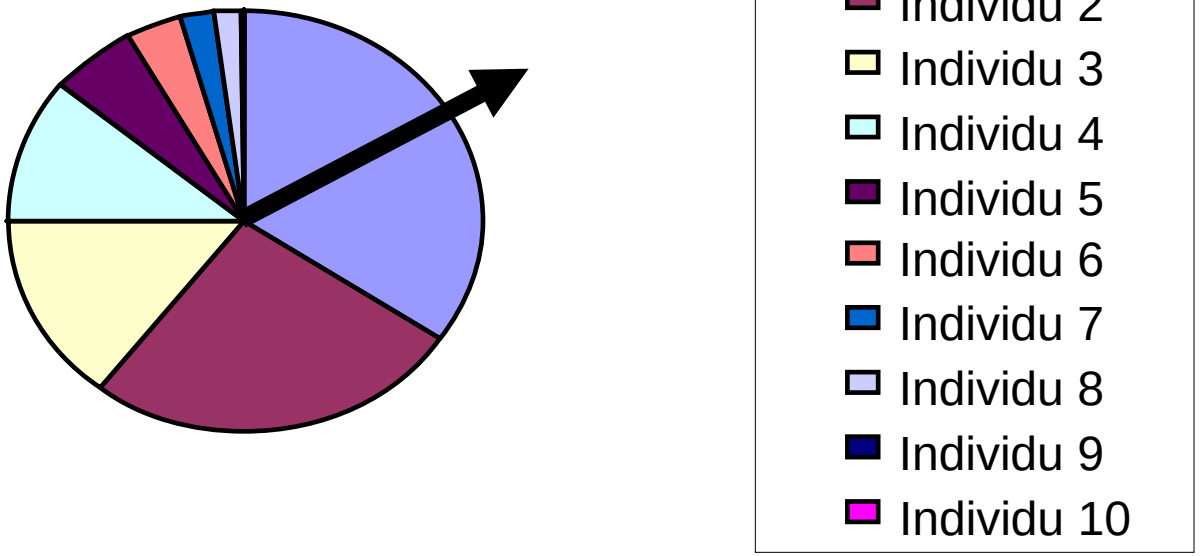
= 2

Fitness Landscape (paysage évolutif) : fonction performance sur l'espace de recherche



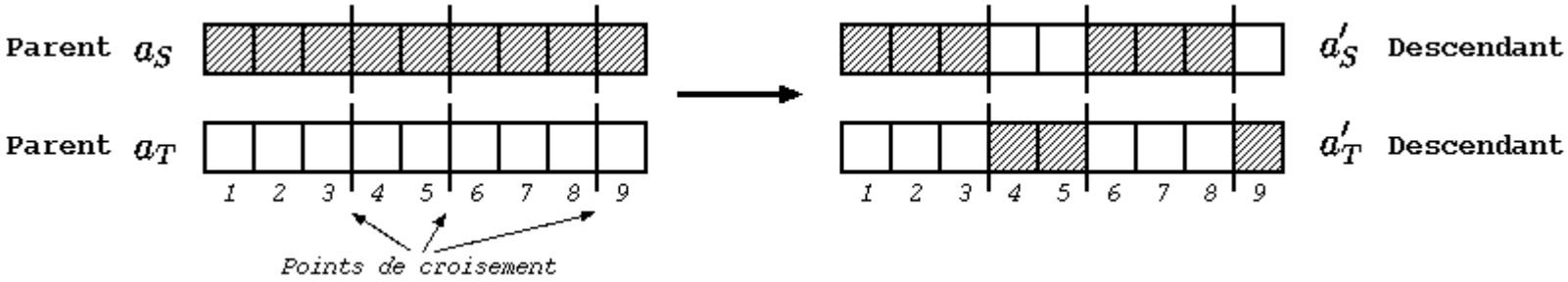
Sélection proportionnelle

- Un individu i de fitness f_i a une probabilité $f_i / \sum_j f_j$ d'être sélectionné comme parent
- Principe de la roulette biaisée
  - Chaque individu reçoit une portion de roulette proportionnelle à sa fitness

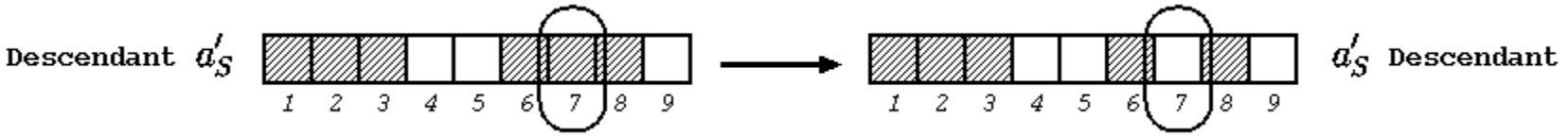


AG : génération

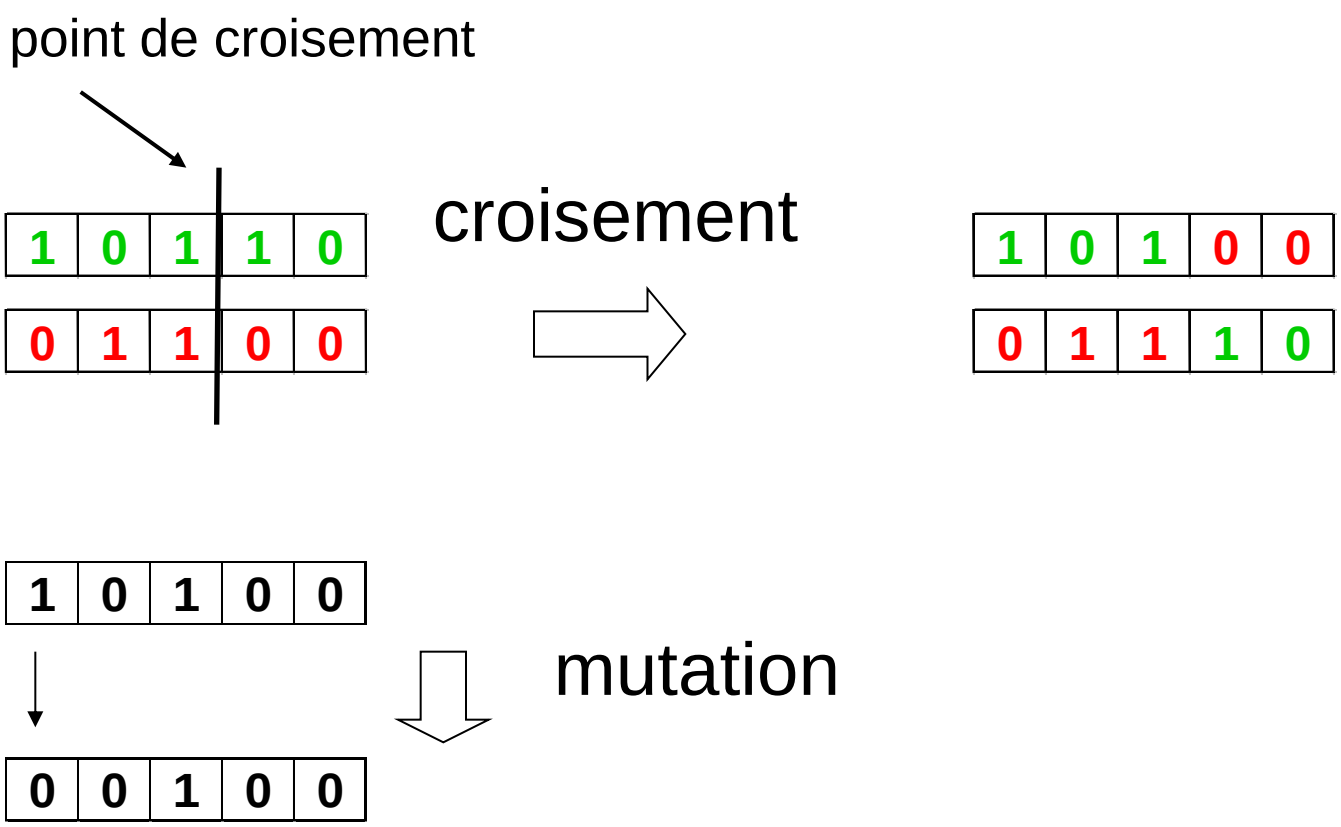
- Sélection
- Croisement / recombinaison (*crossover*)
  - choix des individus formant la nouvelle population



- Mutation
  - modification aléatoire d'un individu



Croisement / mutation



Étapes de mise en œuvre d'un AG

- Choix « algorithmiques »
  - Représentation des individus (codage du génotype),
  - Génération de la population initiale
  - Modalités de reproduction (opérateurs d'évolution)
  - Choix du mode de sélection des individus (opérateurs de sélection)
- Choix « paramétriques »
  - Taille de la population, taux de reproduction, de survie, de croisement, de mutation

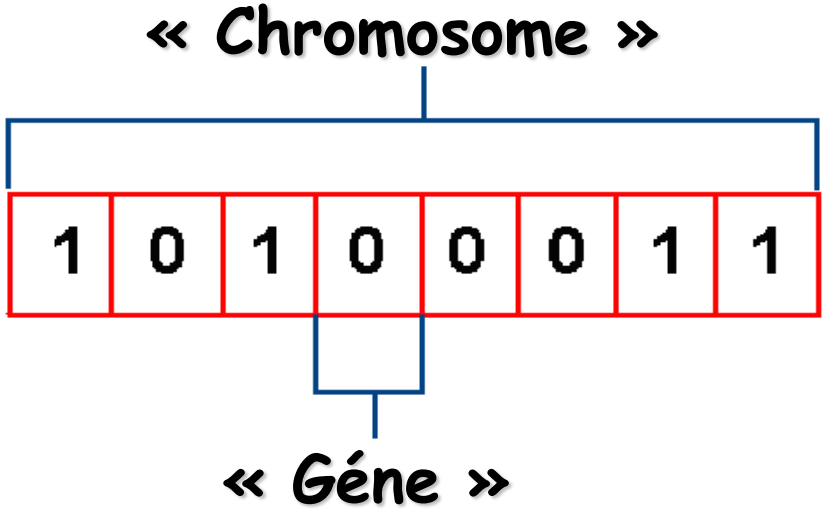
Choix de la représentation (génotype)

- Comment représenter un individu (phénotype ou solution candidate) par un génotype?
- Nombreuses possibilités, **mais**, le codage doit être adapté au problème posé
- Choix de représentation dépend des autres étapes
  - génotype sera converti en phénotype et évalué
  - génotype manipulé par des opérateurs génétiques (mutation, cross-over, …)

↘ *Un codage inadapté peut compromettre l'évolution …*

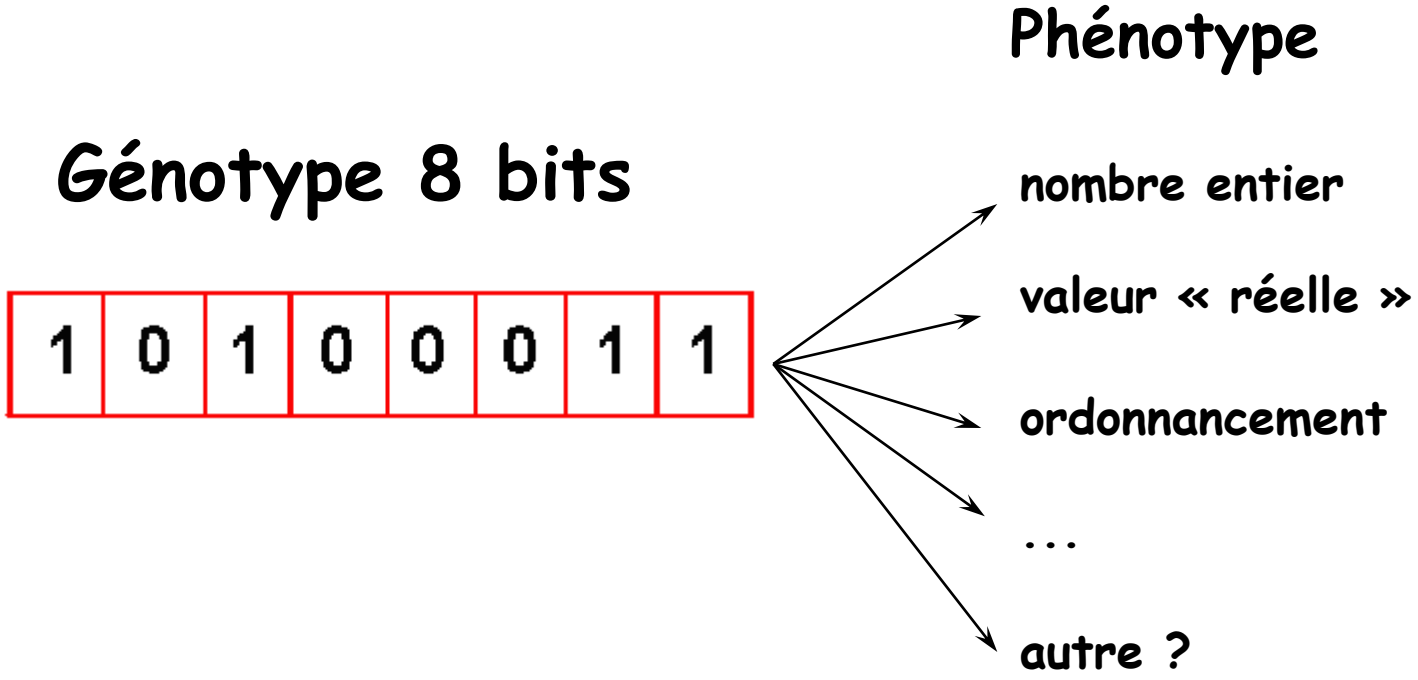
Représentations discrètes

- Un individu peut être représenté par un ensemble de valeurs discrètes (codage binaire, entiers, …)
- **Semble** très proche de la biologie (ACTG) mais …
- Exemple : représentation binaire



Représentations binaires

- Un génotype binaire peut conduire à une grande variété de phénotypes :



Codage de « réels » à partir d'une représentation binaire

- Le codage de valeurs « réelles » est trivial si on connaît la dynamique des paramètres et la précision souhaitée
  - Codage d'un « réel » entre -0.5 et 0.5 avec une précision de 0.001
  - Codage sur 10 bits ([0-1023]) et normalisation ((*x*/1023)-0.5)
- Le codage binaire pose le problème des « falaises de hamming » (*hamming cliffs*)
  - Certaines transitions sont plus probables que d'autres (car le processus de mutation est ponctuel)
  - Cette probabilité n'est pas liée à la proximité des phénotypes mais à la proximité des génotypes (distance de hamming)

⇘ *Un codage inadapté peut compromettre l'évolution …*

Code de Gray (*Gray Coding*)

- Codage qui permet à 2 entiers adjacents d'être distants d'une seule mutation (l'inverse n'est pas vrai !)
  - Pas de falaises de Hamming

- Recette de traduction
  entier binaire => Gray
  - commencer à gauche copier le 1er bit
  - ensuite écrire 1 si le bit change et 0 sinon

Exemple, nombre sur 3 bits :

Binaire	Valeur	Gray
000	0	000
001	1	001
010	2	011
011	3	010
100	4	110
101	5	111
110	6	101
111	7	100

Code de Gray

- Frank Gray, Bell-Labs 1953 pour l'électromécanique, éviter des commandes transitoires pour des ensembles de relais
- Deux entiers adjacents ne diffèrent que de un bit
  - Deux phénotypes proches auront des génotypes proches
- La conversion bit par bit du binaire au code Gray
  - Chaine binaire : $b_1, b_2, \ldots, b_n$
  - Chaine gray : $g_1, g_2, \ldots, g_n = b_1,\ b_1 \oplus b_2, \ldots, b_{n-1} \oplus b_n$

Entiers :	0	1	2	3	4	5	6	7
Binaire :	000	001	010	011	100	101	110	111
Gray :	000	001	011	010	110	111	101	100

Cas particulier: représentation ordonnée

- Un individu est représenté par une liste de positions avec permutations
- Application : ordonnancement / séquencement
- Exemple : Le voyageur de commerce
  - Chaque ville reçoit un numéro d'ordre de 1 à *n*
  - Les solutions sont représentées par un ordre (ex : 5, 4, 2, 1, 3).

⇘ *Les représentations ordonnées doivent être utilisées avec des opérateurs génétiques spécifiques assurant la validité des individus*

Représentation à valeurs réelles

- Si la solution recherchée est exprimée sous la forme d'une liste de valeurs réelles, le codage le plus naturel est d'exprimer le génome sous la forme d'une liste de réels ! (abstraction faite du codage binaire)
- Cas très courant couvrant un grand nombre d'applications (optimisation paramétrique)

↘ *La manipulation individuelle, par des opérateurs génétiques, des bits d'un codage flottant n'a aucun sens ...*

Représentation à valeurs réelles

- Un individu X est un vecteur à valeurs réelles :

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x_i \in R$$

- La fonction de fitness f permet d'exprimer la qualité d'un individu (fonction de R^n dans R)

$$f : R^n \rightarrow R$$

Initialisation de la population

- *Si possible*: tirage aléatoire uniforme dans l'espace de recherche
  - Codage binaire : 0 ou 1 avec une probabilité de 0.5
  - Codage à valeurs réelles : tirage uniforme dans un intervalle donné (applicable uniquement dans un ensemble borné)
  - Attention : un tirage uniforme des génotypes ne garantit pas toujours un tirage uniforme des phénotypes

Evaluation des individus

- Pour les applications réelles, cette étape est - de loin - la plus coûteuse
  - Difficulté de conception, temps de calcul, prix …
  - Ne pas évaluer de nouveau les individus identiques ou non modifiés lors de leur reproduction
- Plusieurs méthodes en fonction du problème :
  - Évaluation directe (fonction de fitness),
  - Simulation (boite noire),
  - Processus externe (robot, tests en situation, …)

    ↘ *Pour accélérer le processus il est possible d'utiliser une évaluation approchée … au début du processus d'évolution …*

Cas particuliers de l'évaluation

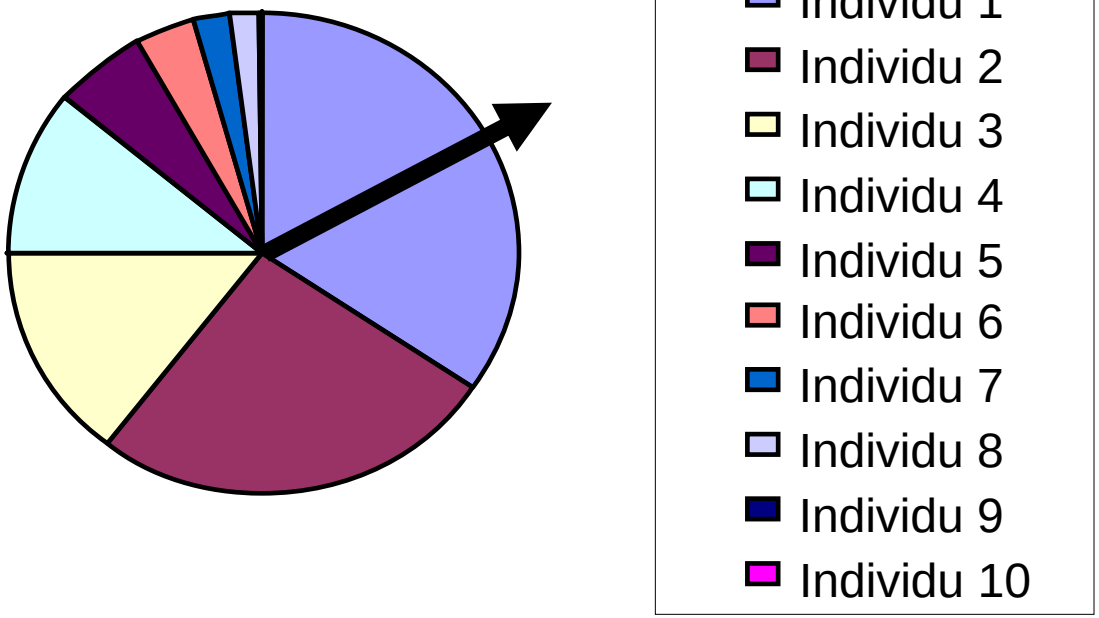
- Satisfaction de contraintes : que faire lorsque le phénotype rompt des contraintes du problème ?
  - Pénaliser l'individu (impact sur le fitness)
  - Utiliser des opérateurs génétiques spécifiques

Stratégie de sélection « naturelle »

- But de la sélection naturelle :
  - Assurer que les meilleurs individus aient plus de chance d'avoir une descendance que les autres.
- La « pression de sélection » doit guider l'exploration
- Les moins bons individus doivent conserver une chance de se reproduire.
  - ils peuvent contenir des gènes intéressants.
  - => la pression de sélection doit préserver une part de bio-diversité.

Sélection proportionnelle

- Un individu i de fitness f_i a $n.(f_i / \sum_j f_j)$ chances de se reproduire (n étant le nombre d'individus sélectionnés).
- Principe de la roulette biaisée.
  - Chaque individu reçoit une portion de roulette égale à sa fitness
  - La roulette est tirée n fois



Sélection proportionnelle

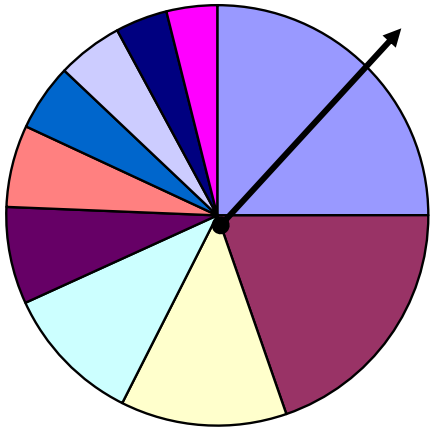
- Désavantages de la sélection proportionnelle :
  - Réduction rapide de la diversité
  - Risque de convergence prématurée (les individus les mieux adaptés vont rapidement phagocyter l'ensemble de la population)
  - La pression de sélection reste faible lorsque les fitness sont très similaires
  - Plusieurs solutions pour éviter la convergence prématurée (vers un optimum local)

⇘ *Normalisation de la fonction de fitness …*

Sigma Scaling

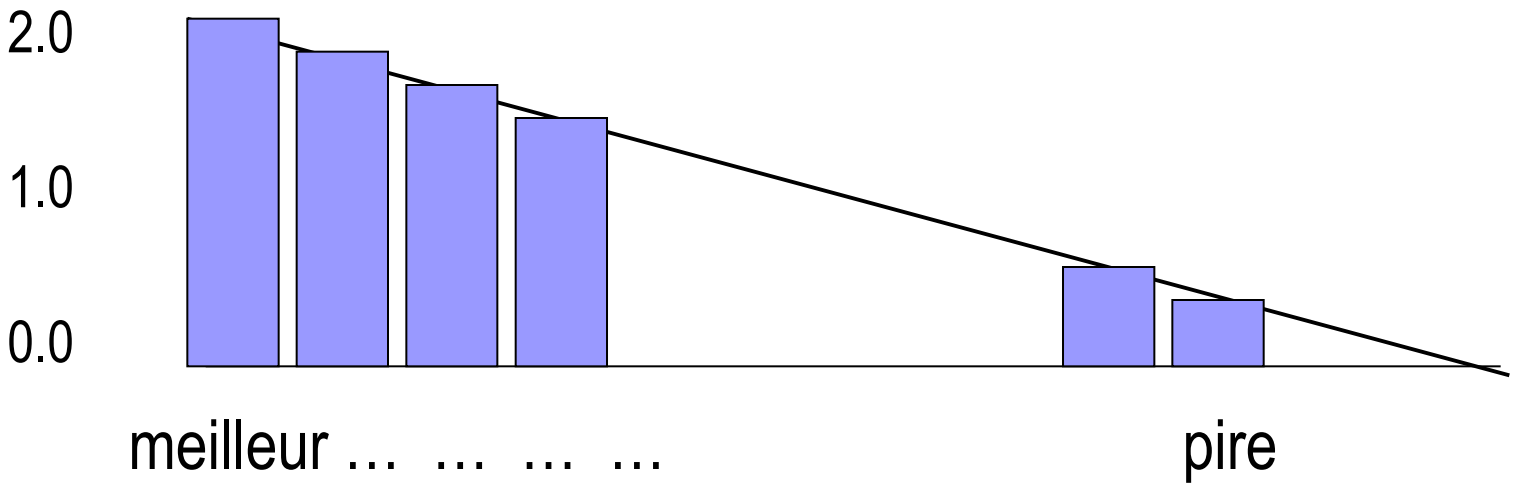
- But : éviter la perte de biodiversité si un individu est beaucoup mieux adapté que les autres
  - La grande majorité des descendants sont issus d'un seul individu
  - Risque de convergence prématurée
- Normalisation basée sur la variance (sigma scaling)
  - Uniformisation de la pression de sélection

$$\begin{cases} Si\ \sigma \neq 0 & Fit(x) = 1 + ((f(x) - Mean(f)) / 2*\sigma) \\ Sinon & Fit(x) = 1 \end{cases}$$



Sélection par le rang

- Sélection linéaire par le rang aligne la population selon le rang et donne une probabilité de sélection proportionnelle au rang

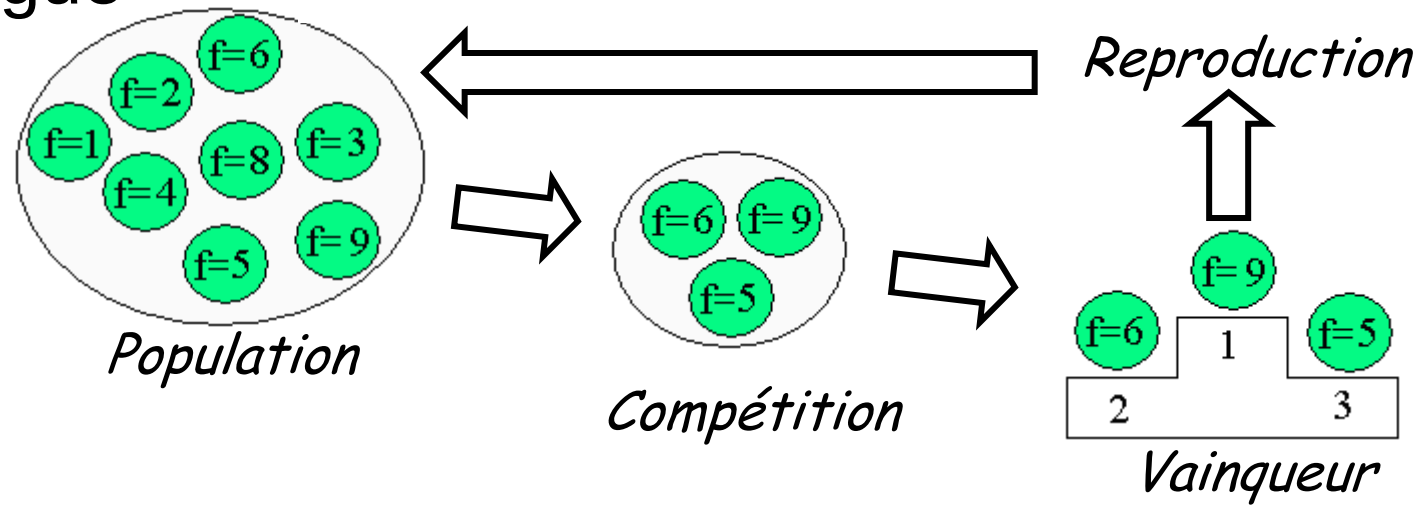


Sélection par le rang

- La droite ne descend pas obligatoirement entre 2.0 et 0.0, par exemple entre 1.5 et 0.5
- Possibilité d'un rang non linéaire
- Plus couramment utilisé: rang linéaire avec une pente de 2.0 à 0.0
- Ainsi le **meilleur** se reproduit **2 fois** plus que le **médian**. Les solutions inférieures à la moyenne gardent une chance de se reproduire

Sélection par compétition (*tournament*)

- Sélection basée sur un remplacement partiel de la population
  - Sélection aléatoire de k individus (sans remplacement),
  - Compétition entre les k individus,
  - Seul le meilleur se reproduit
- Très en vogue



Stratégie de « remplacement » (mortalité)

- Outre le critère de sélection, la pression de sélection est fonction de la « mortalité » des individus
- En AG, la mortalité est généralement liée au « remplacement » (population constante)
- Deux approches :
  - Stratégies stochastiques
  - Remplacement déterministe

⇘ *Élitisme : conservation systématique du (des) meilleur(s) individu(s) …*

Autres stratégies de remplacement

- Si N parents génèrent N enfants et sont tous remplacés, la sélection se fait uniquement sur le choix des parents
- Mélanger parents et enfants puis sélectionner pour le remplacement
- Générer plus d'enfants puis les sélectionner

Intérêt de l'élitisme

- Comment assurer une croissance permanente du fitness ?
  - Élitisme : ré-introduire le ou les meilleur(s)
  - Préservation : « niche » pour le meilleur individu
- L'élitisme présente aussi un risque
  - Convergence prématurée due à la conservation d'un individu qui « pollue » en permanence la population

⇘ *Intérêt principal de l'élitisme : éviter la frustration de l'utilisateur …*

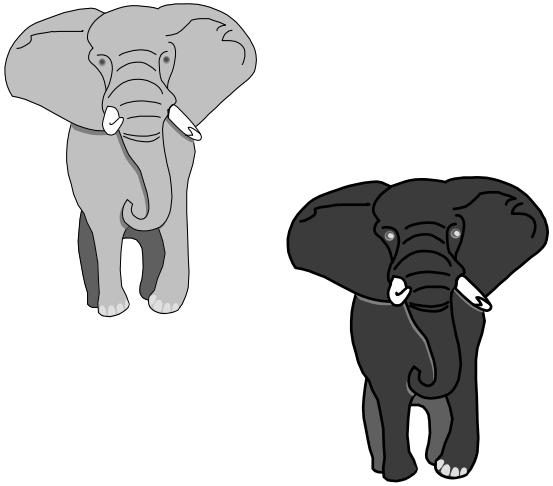
Les opérateurs génétiques : la mutation

- Il peut y avoir un ou plusieurs opérateurs de mutation, mais :
  - un des opérateurs de mutation au moins doit permettre d'atteindre tous les points de l'espace de recherche
  - le taux de mutation est un facteur fondamental qui doit pouvoir être finement contrôlé
    - ⇘ *Les mutations doivent produire des chromosomes (i.e. des individus) valides (i.e. viables)*

Mutation pour une représentation binaire

Génotype

Phénotype



Avant 1 0 1 1 1 1 1

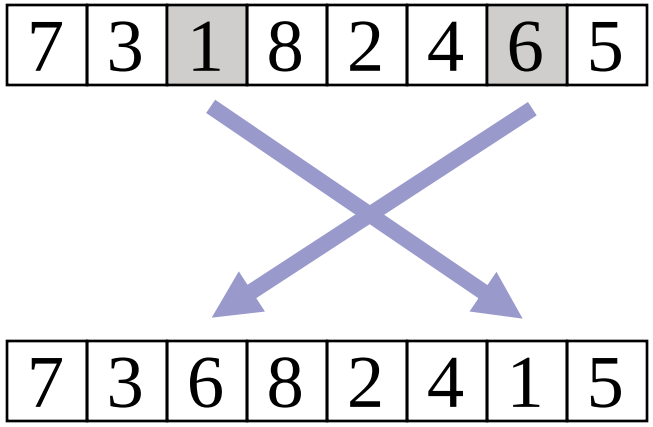
Après 1 0 1 0 1 1 1

Gène mutant

↘ *Les mutations apparaissent avec une probabilité* p_m *pour chaque gène (taux de mutation).*

Le cas des représentations ordonnées

- Échanger deux gènes choisis aléatoirement (*gene swap*)



Le cas des génomes à valeurs réelles

- Perturbation des gènes par addition d'un bruit aléatoire
- En général on utilise un bruit Gaussien obtenu par une loi normale $N(0,\sigma)$
  - 0 est la valeur moyenne du bruit
  - σ est l'écart type
- Soit, pour chaque paramètre : $x'_i = x_i + N(0,\sigma_i)$
- L'utilisation d'un bruit à distribution uniforme est possible

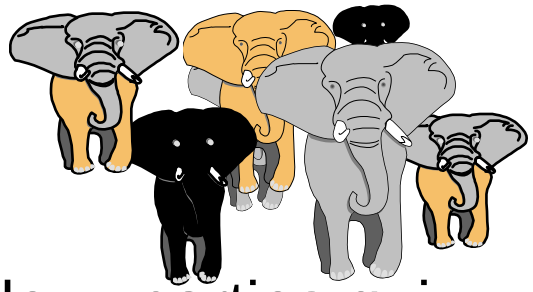
Les opérateurs génétiques: le cross-over

- Il peut y avoir un ou plusieurs opérateurs de recombinaison (cross-over), mais :
  - les descendants doivent hériter de chacun des parents (sans quoi il s'agit d'une simple mutation)
  - les opérateurs de recombinaison doivent tenir compte de la représentation du génotype et de la fonction de transfert génotype/phénotype
- Le cross-over est souvent appelé «Crossing-over» ou «recombinaison»
  - En particulier en biologie

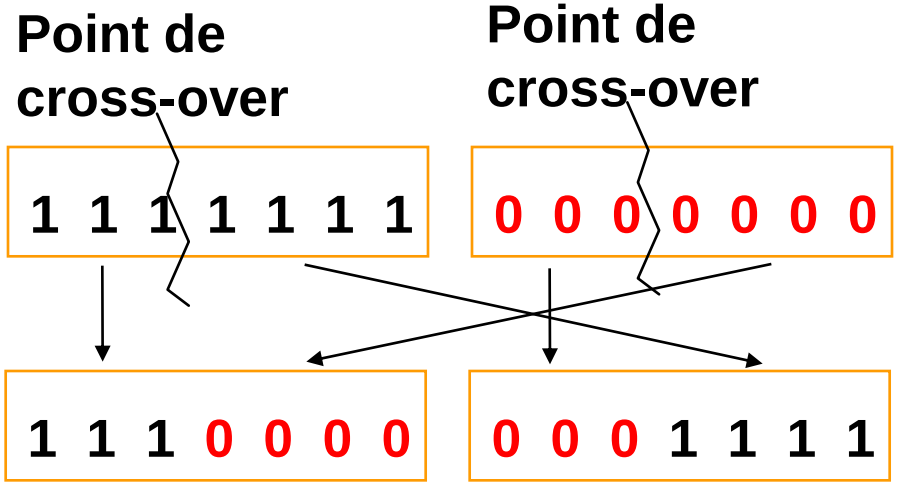
  ⇘ *Les recombinaisons doivent produire des chromosomes (i.e. des individus) valides (i.e. viables)*

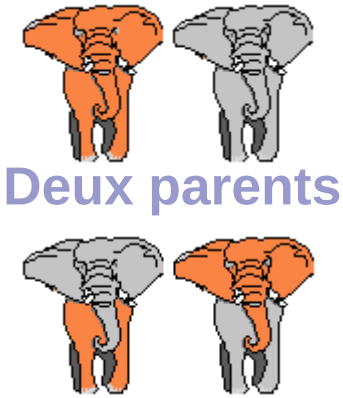
Le cas des représentations binaires

- Deux parents donnent deux enfants
  - Parfois un seul



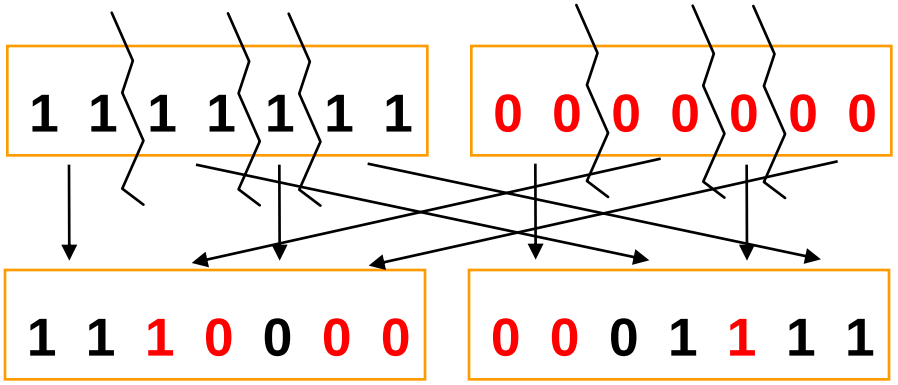
- Chaque chromosome est découpé en deux parties qui sont recombinées pour former les descendants





Variantes pour le cross-over discret

- Cross-over multipoints
  - *k* points de cross-over tirés au hasard

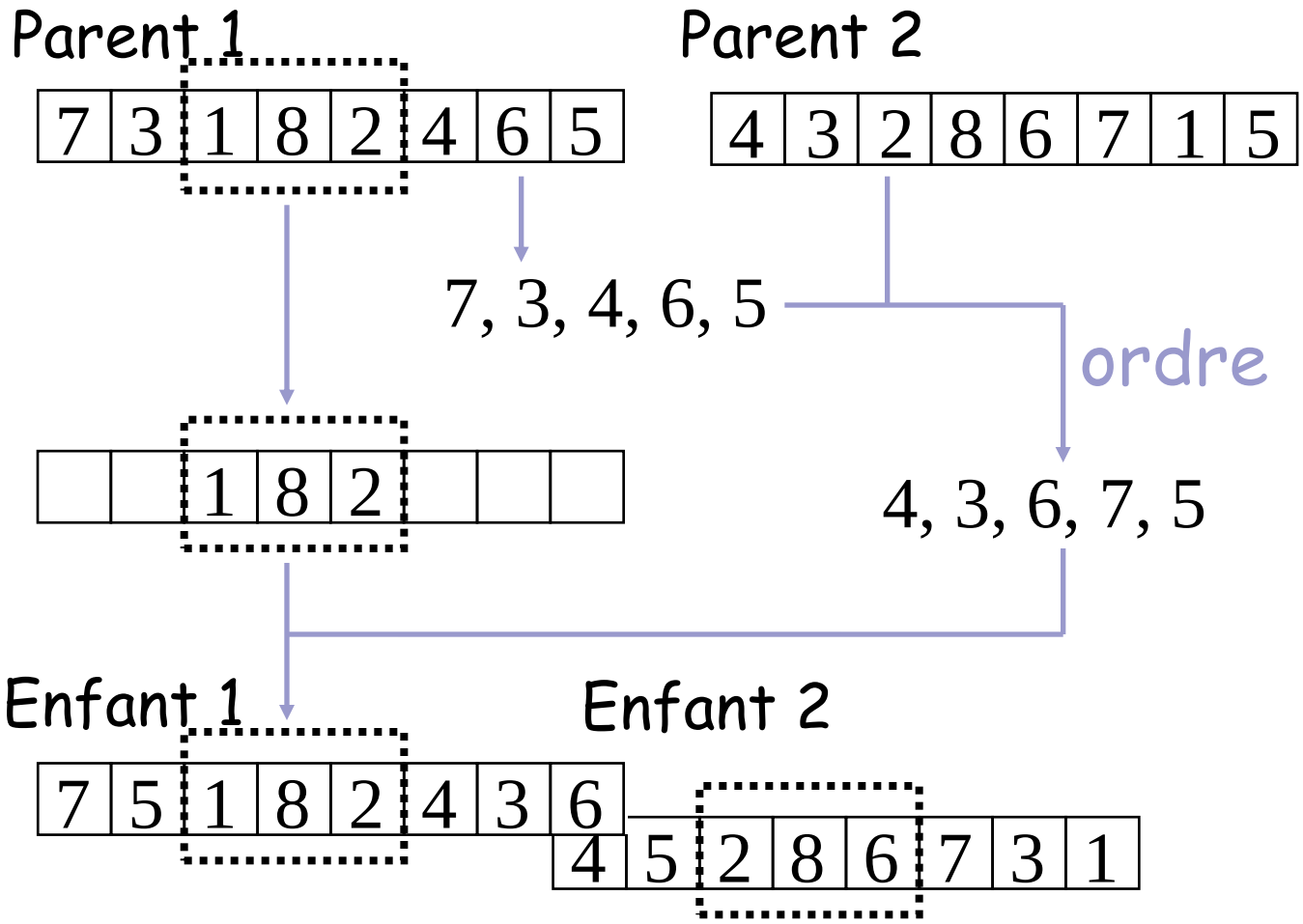


- Cross-over uniforme
  - Les gènes sont échangés suivant une probabilité p_c

Le cas des représentations ordonnées

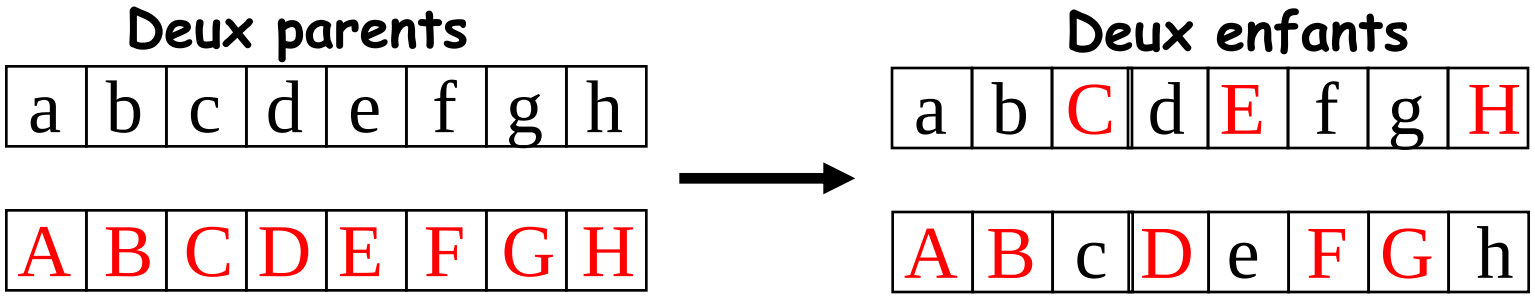
- Choisir aléatoirement une partie du génome du premier parent et le recopier à l'identique
- Compléter le génome du descendant avec le deuxième parent
  - à partir du point suivant la partie recopiée
  - utiliser l'ordre des gènes du deuxième parent
  - balayage circulaire du chromosome
- Inverser le rôle des parents pour créer un deuxième descendant

Le cas des représentations ordonnées



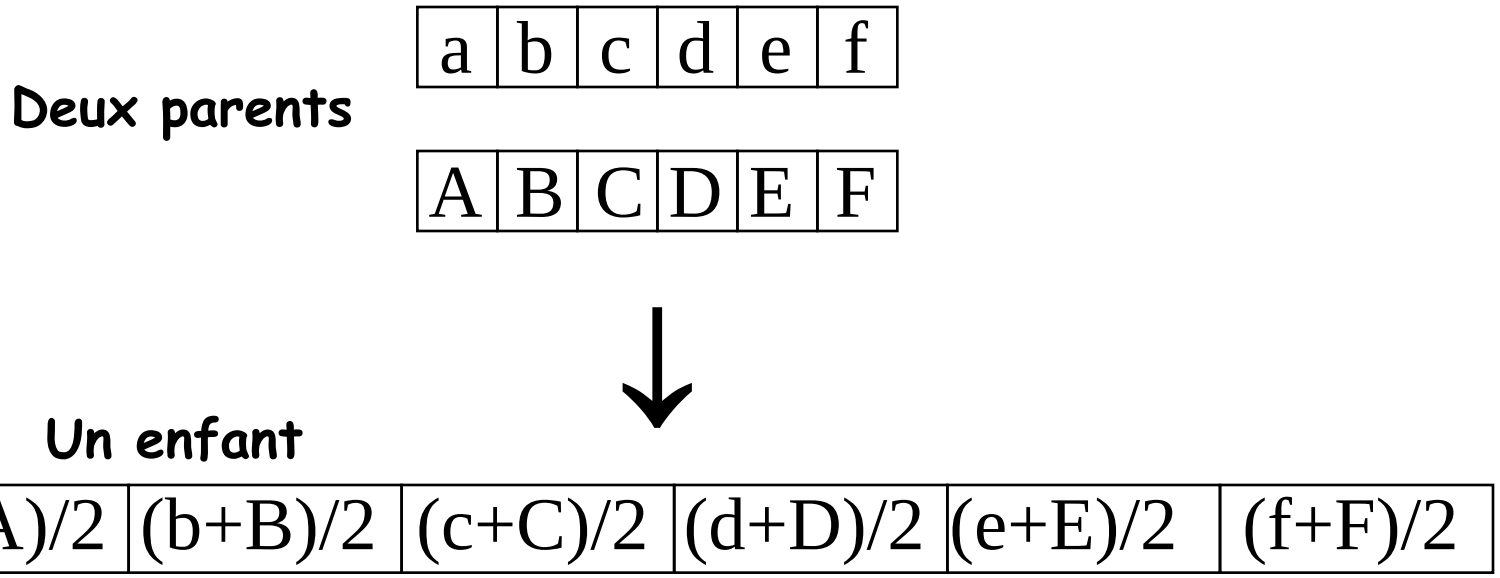
Le cas des génomes à valeur « réelle »

- Recombinaison discrète :
  - Recombinaison aléatoire des gènes à partir des deux parents
  - Généralement pas de contraintes topologiques sur le génome



Le cas des génomes à valeur « réelle »

- Recombinaison discrète :



Intérêts relatifs de la mutation et du cross-over

- Cross-over
  - Faible part d'aléa, permet de ré-exprimer des caractères déjà présents dans la population,
  - L'effet du cross-over diminue lorsqu'on se rapproche de la convergence,
  - Opérateur d'***exploitation***
- Mutations
  - Indispensables pour éviter les maxima locaux,
  - Opérateur d'***exploration*** ...

↘ *La recombinaison est souvent plus difficile à mettre en œuvre ...*

Intérêts relatifs de la mutation et du cross-over

- L'intérêt du cross-over varie beaucoup suivant les problèmes :
  - La fonction de fitness est-elle fiable ?
  - Les opérateurs de recombinaison ont-ils un sens compte-tenu du codage utilisé ?

  ↘ *Il est possible de « mixer » mutation et cross-over en autorisant la parthénogenèse (recombinaison d'un individu avec lui-même) et la reproduction sexuée*

Critère d'arrêt ...

- Difficile à déterminer puisque la qualité de la solution s'améliore de façon quasi-continue
  - Plusieurs « solutions »
- Le maximum est atteint
  - Suppose qu'il soit connu !
  - Il s'agit souvent d'un maximum *utile ...*
- Limite de temps de calcul
  - On ne peut pas toujours évaluer autant d'individus qu'on le souhaite
- Limite de patience de l'utilisateur
  - Lorsque rien n'évolue pendant plusieurs générations

Choix paramétriques ...

- Outre les choix « algorithmiques », la mise en œuvre d'un algorithme génétique demande de fixer un certain nombre de paramètres
- n : nombre d'individus dans la population

  de l'ordre de la centaine (souvent fonction de la difficulté de calcul de la fonction de fitness)
- T_m : taux de mutation

  de l'ordre de 1/1000 (fonction de l'influence des différents gènes ou de l'ordre de 1 mutation par génome
- T_c : taux de cross-over

  de l'ordre de 1/2 (fonction de l'intérêt/du sens du cross-over)

Performance des algorithmes génétiques

- Du point de vue théorique :
  - Ne jamais tirer de conclusion à partie d'un test unique !
  - Les performances doivent être estimées statistiquement
  - Donc à partir d'un grand nombre d'essais

Point clef de l'algorithmique génétique

- Maintien de la diversité génétique au cours de l'évolution
  - Il faut maintenir des caractéristiques génétiques différentes dans la population (bio-diversité)
  - Lorsqu'on perd la diversité génétique, tous les individus deviennent semblables
  - Effet boule de neige
  - Convergence vers l'optimum local le plus proche

  ⇘ *En théorie, les mutations permettent de continuer à explorer l'espace … en pratique, la perte de diversité génétique est irréversible …*

Point clef de l'algorithmique génétique

- Dilemme Exploration / Exploitation
  - Exploration = tester les zones inconnues

    Trop d'exploration revient à une marche aléatoire et compromet la convergence
  - Exploitation = essayer d'améliorer le meilleur individu trouvé jusqu'ici

    Trop d'exploitation revient à une recherche locale et conduit à une convergence vers un optimum local

  ⇘ *Dilemme exploration / exploitation*

Avantages des AGs

- Bon rapport coût / résultat sur une grande classe de problèmes
- Parallélisme intrinsèque
- Robuste, tolérant aux fautes
- Applicable sans connaissance préalable du domaine d'application
- Simple à programmer
- Attention, « everything is problem dependent »
  - Les algorithmes génétiques ne sont pas **la** panacée universelle
- les AGs sont particulièrement adaptés lorsque le problème :
  - contient beaucoup de données / de paramètres
  - contient des paramètres interdépendants (problème complexe)
  - comporte des optimums locaux

Inconvénients des AGs

- Pas de garantie de convergence en un temps fini
- Faiblesse des fondements théoriques et mathématiques
- Souvent gourmands en calcul donc lents (mais aisément parallélisables)
- Chaque individu doit être évalué, même les individus inadaptés (utilisation offline, sur simulateur)
- Produit toujours des individus inadaptés

⇘ *Le comportement de l'algorithme dépend d'un grand nombre de paramètres qui peuvent être difficiles à fixer*