جامعة 8 ماي 1945 قالمة
UNIVERSITE 8 MAI 1945 - GUELMA

LabSTIC
Université 8 Mai 1945

Les approches évolutionnistes

Dr. Mohammed Nadjib KOUAHLA

Motivation

Source d'inspiration : Ils sont basés sur **la théorie de l'évolution** des espèces (Darwin), selon laquelle les gènes conservés d'une génération à l'autre sont ceux les plus adaptés à l'espèce dans son environnement,

- **Dès 1962**, **Dr John Henry Holland** et ses travaux sur les systèmes adaptatifs : ***Crossing-Over*** en complément des mutations

- **Années 1990**, **vulgarisation** des algorithmes génétiques avec la publication de **David Golberg**

Motivation

- **3 Types d'Algorithmes Évolutionnaires**, aujourd'hui regroupés:
  - *Algorithmes Génétiques*
  - *Stratégies d'Évolution*
  - *Programmation Évolutionnaires*

Notons également les domaines de la *Programmation Génétique* et de la *Vie Artificielle.*

Types d'algorithmes évolutionnaires

- **Stratégies d'Evolution (SE) :** les (SE) ont été développés par deux ingénieurs Rechenberg et Schwefel au cours de leurs travaux de recherches sur des problèmes numériques pour l'optimisation paramétrique.

- **Programmation Evolutionnaire (EP) :** la (PE) a été développée aux années 60 par L. Fogel dans le contexte de la découverte d'automates d'états finis pour l'approximation des séries temporelles, puis utilisé plus tard par D.B. Fogel aux années 91 .

- **Algorithmes Génétiques (AG) :** les (AG), développées par J. Holland en 75 comme outils de modélisation de l'adaptation et qui travaillent dans un espace de chaînes de bits. Les plus populaires aux chercheurs de différentes disciplines. Largement utilisées et développées par D.E. Goldberg en 1989.

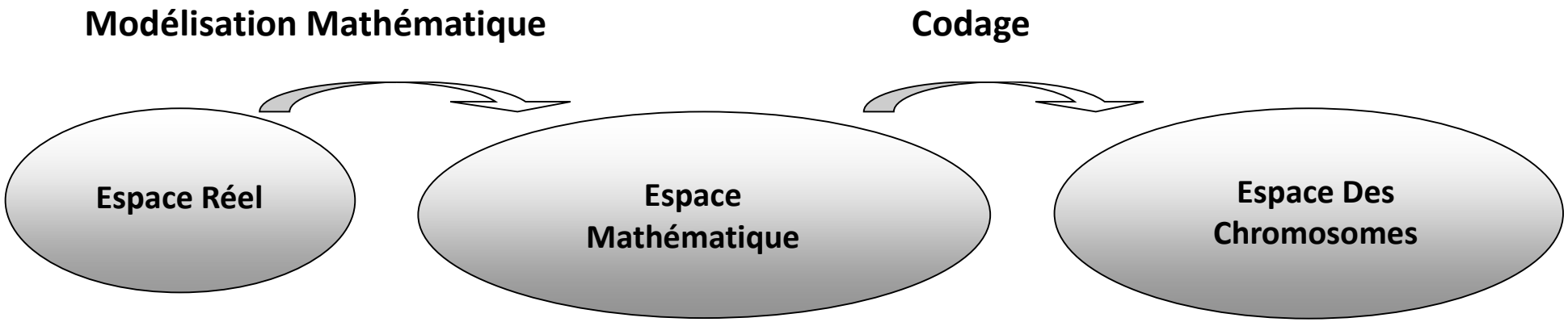
- **Programmation Génétique (PG) :** la (PG) a été développée par J. Koza en 90 dans le but d'attendre un des rêves des programmeurs : faire écrire un programme automatiquement. La représentation est basée sur des arbres représentant les programmes.

Les algorithmes génétiques

Principe:

Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants :

1. Un principe de **codage** de l'élément de population. Dans cette étape on associe à chaque point de l'espace une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. Le choix du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très employés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs, pour l'optimisation de problèmes à variables continues.



Principe:

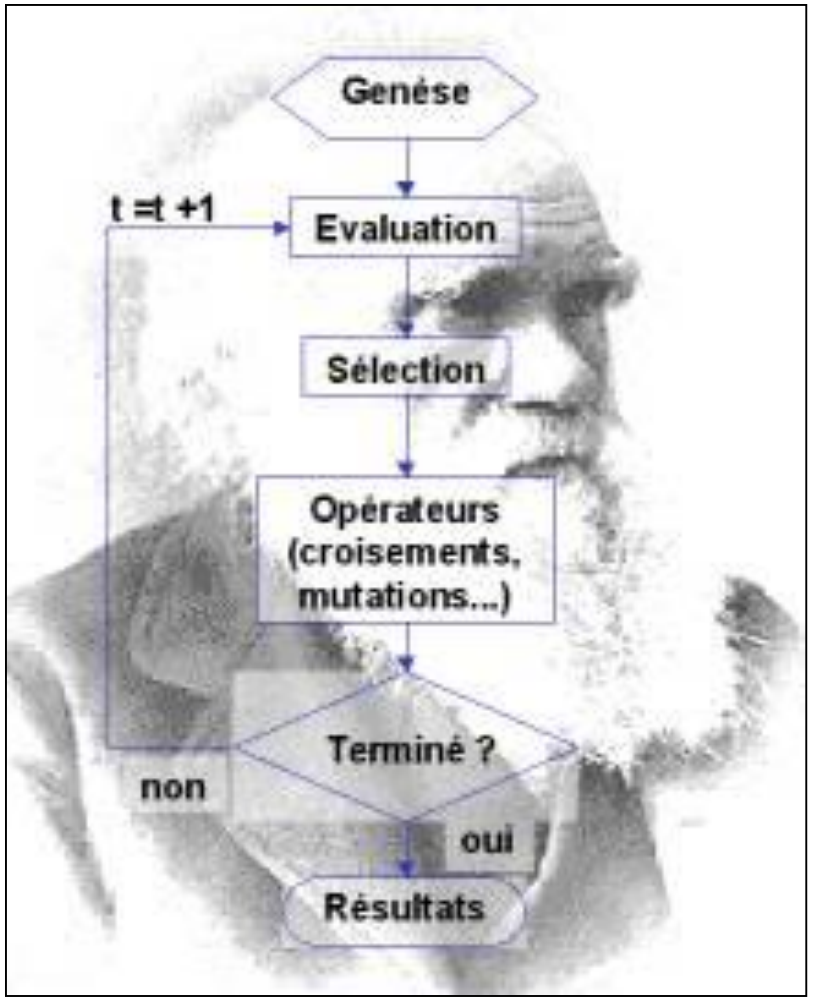
2. Un mécanisme de **génération de la population initiale**. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

3. Une **fonction à optimiser**. Celle-ci prend ses valeurs dans R+ et est appelée *fitness* ou fonction d'évaluation de l'individu. Celle-ci est utilisée pour sélectionner et reproduire les meilleurs individus de la population.

4. Des **opérateurs** permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.

5. Des **paramètres de dimensionnement** : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

Organigramme



Vocabulaire

Population

- Individus
- Gènes
- Chromosomes
- Mutations
- Parents
- Descendants
- Reproduction
- Croisements

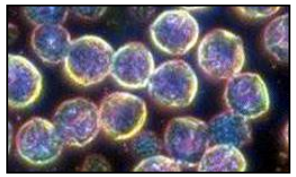
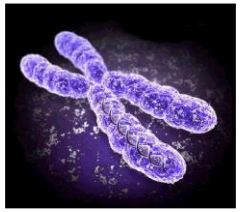
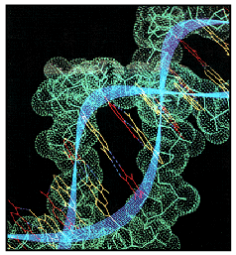
Analogies avec des **phénomènes biologiques**

Vocabulaire

Algorithmes issues de la biologie : Génétique

Un **Individu** est composé de:

Cellules → Chromosomes → ADN

- ADN = Chaîne de **Gènes**
- Variantes d'un **Gène** = **Allèle**
- Emplacement du **Gène** sur le Chromosome = **Locus**
- Ensemble des Chromosomes = **Génome**

Vocabulaire

Les Outils :

- Sélection (sélection naturelle)
  → Amélioration **globale** de l'adaptation
- **Recombinaison (*crossing-over*)**
  → **Opération prépondérante**, simple ou multiple
- Mutation
  → Pas de **convergence prématurée**, minimums et maximums locaux

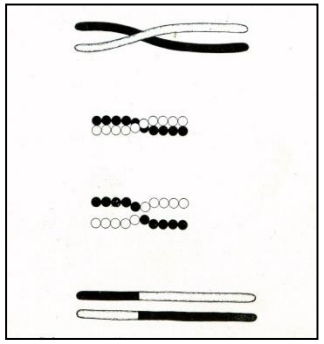
Vocabulaire

Différents types de sélection:

- Par rang (élitiste)
- Roue de la fortune (roulette)
- Par tournoi
- Uniforme

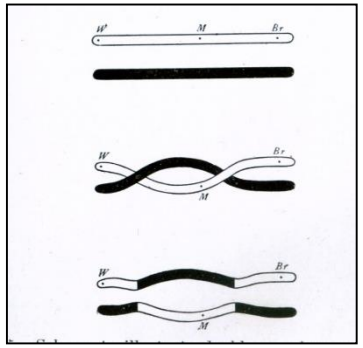
Recombinaison (crossing-over)

Simple



Chromosome	Contenu
A	00 : 11 00 10
B	01 : 01 01 00
A'	00 : *01 01 00*
B'	01 : *11 00 10*

Multiple



Chromosome	Contenu
A	00 : 11 00 : 10
B	01 : 01 01 : 00
A'	00 : *01 01 : 10*
B'	01 : *11 00 : 00*

Les mutations :

Taux relativement faible et évolutif

- Permet d'éviter les problèmes d'optimums locaux

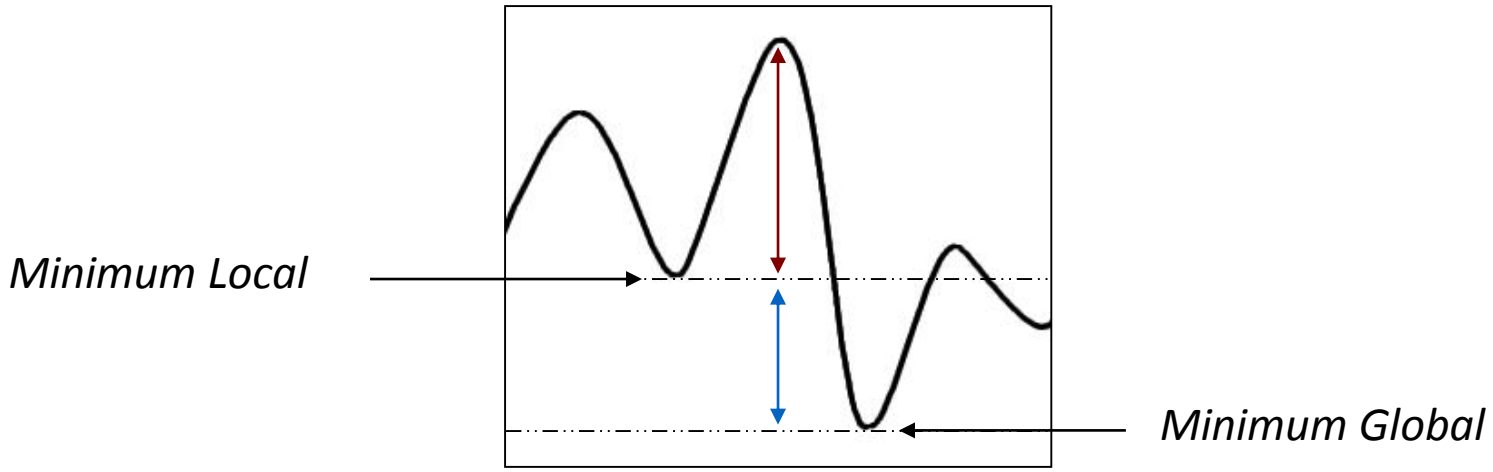
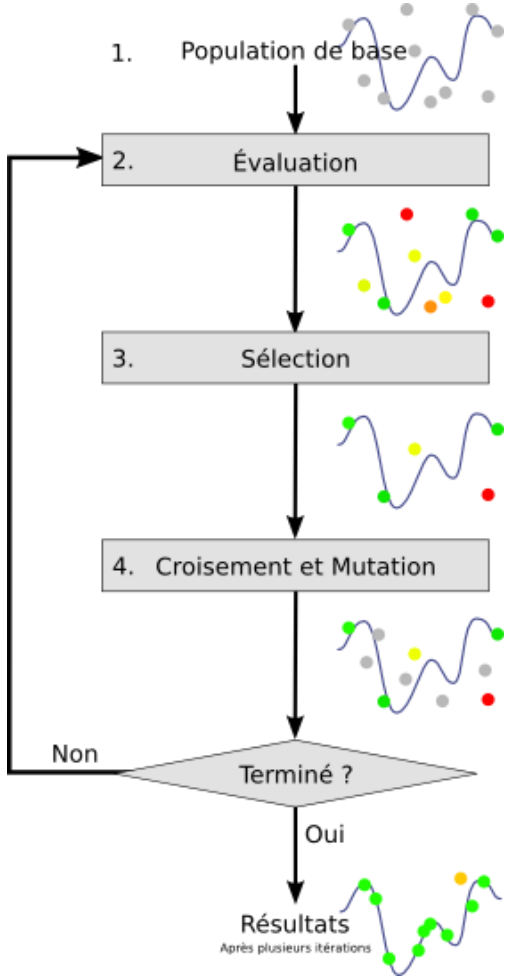


Schéma Récapitulatif

Cycle qui se répète jusqu'à la **condition d'arrêt** :

- Nombre de **générations fini**
- **Score** des Individus



Domaines d'application

Applications **multiples** :

→ **Optimisations** de fonctions numériques difficiles, d'emplois du temps, de design, **traitement d'image**, contrôle de systèmes industriels …

Les Algorithmes Génétiques peuvent être utilisés pour contrôler un **système évoluant dans le temps** :

→ **Adaptation** de la population à des **conditions changeantes**

Questions de modélisation

1. Définition d'une fonction d'adaptation (fitness) :

C'est la fonction à optimiser.

- Son expression peut être immédiate et assez simple (fonction mathématique), mais ceci n'est pas toujours le cas. On peut donc être amener à en considérer une approximation.
- A définir en fonction du problème

Questions de modélisation

2. Choix du codage des chromosomes :

chromosome:={A}$_l$

- {A} alphabet binaire ou non
- l: longueur du chromosome (fixe ou variable)

Buts

- pouvoir coder toutes les solutions
- faciliter la mise en œuvre des opérations de reproduction.

Choix en fonction du problème.

En général on distingue deux classes de problèmes :

- les problèmes à valeur: on cherche les meilleures valeurs possibles des gènes,
- les problèmes de position: l'ensemble des valeurs finales est connu et on cherche la meilleure position pour placer chacune de ces valeurs.

Questions de modélisation

3. Définition des opérations de reproduction

- **Croisement** :
  - S'applique sur deux individus différents.
  - Résultat: chromosome formé à partir des gènes de ses deux parents.
  - Deux enfants sont " produits " pour la génération suivante.
  - Un pourcentage de croisement est fixé.
- **Mutation** :
  - S'applique sur un seul individu par la modification de l'un ou plusieurs gènes du parent choisi(s) aléatoirement.
  - Un seul nouvel enfant est fourni.
  - Un pourcentage de mutation est fixé.

Questions de modélisation

- 4. Choix de la méthode de sélection :

  Trois objectifs principaux:
- Elle permet de choisir les individus sur lesquels s'appliqueront les opérations de reproduction pour la création de la future génération.

- Elle doit " favoriser " les meilleurs individus

- Elle doit permettre d'explorer les différentes parties de l'ensemble de recherche

Questions de modélisation

- Chercher les méthodes de sélection
- Quelles sont les avantages et les inconvénients de chacune?

Questions de modélisation

- 5. Convergence (Arrêt du processus)
- De générations en générations la fonction d'adaptation du meilleur chromosome et la moyenne de l'ensemble de la population vont évoluer et tendre vers l'optimum (global).
- La convergence est l'augmentation progressive vers de l'uniformité.
- Un gène a convergé quand 95% de la population possède la même valeur du gène.
- La population a convergé quand tous les gènes ont convergé

La programmation génétique

La programmation génétique

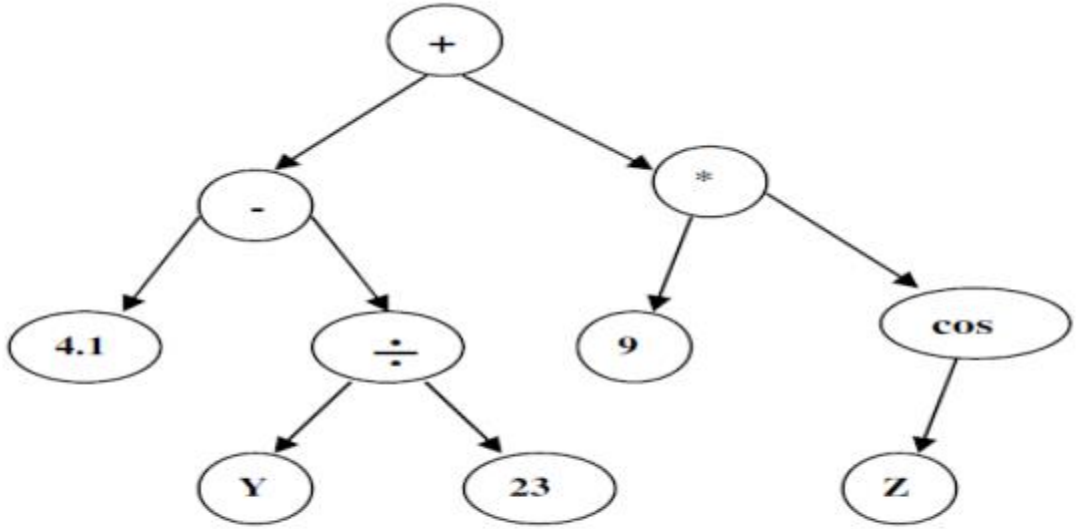
La programmation génétique (Genetic Programming) est une technique d'optimisation évolutionnaire proposé par *John Koza* en 1990. Techniquement, l'algorithme de programmation génétique est une extension de l'algorithme génétique.

L'objectif de l'algorithme de programmation génétique consiste à utiliser l'induction pour élaborer un programme d'ordinateur. Ce résultat est obtenu en utilisant des opérateurs évolutifs sur les programmes candidats avec une structure d'arborescence afin d'améliorer l'ajustement adaptatif entre la population de programmes candidats et une fonction objective. Une évaluation d'une solution candidate implique son exécution.

La programmation génétique fait évoluer des programmes informatiques, traditionnellement représentés dans la mémoire avec des structures arborescentes.

Les arbres peuvent être facilement évalués de manière récursive. Chaque nœud de l'arbre possède une fonction d'opérateur et chaque nœud terminal possède un opérande, ce qui rend les expressions mathématiques faciles à évoluer et à évaluer.

La programmation génétique



$$\left(4.1-\left(\frac{Y}{23}\right)\right)+(9*\cos(Z))$$

La programmation génétique

En général, la programmation génétique développe des expressions symboliques sous un langage fonctionnel, un programme évolutionnaire peut contenir des segments de code qui, lorsqu'il est retiré du programme ne modifierait pas le résultat produit par le programme.

La taille du programme évolutionnaire peut aussi se développer de manière incontrôlable jusqu'à ce qu'il atteigne la profondeur maximale autorisée de l'arbre alors que le degré d'aptitude reste inchangé. Cet effet est connu sous le nom de ballonnement, il s'agit d'un grave problème dans la programmation génétique, car il conduit généralement à la réduction de l'effet des opérateurs de recherche.

Une fois qu'il se produit, le degré d'aptitude baissera fortement.

La programmation génétique : principe

l'algorithme consiste à faire évoluer une population constituée d'un grand nombre de programmes.
Au départ, la population est constituée de programmes créés aléatoirement. Chaque programme est évalué selon une méthode propre au problème posé.

A` chaque itération (génération) on classe les programmes en fonction des notes qu'ils ont obtenues, et on crée une nouvelle population, où les meilleurs programmes auront une plus grande chance de survivre ou d'avoir des enfants que les autres.

Ce principe est le même qu'en algorithme génétique classique, mais les opérateurs de croisement et de mutation sont un peu différents. En effet, ils travaillent directement sur la structure d'arbre du programme.

La programmation génétique

1. Génération aléatoire de la population (1 individu = 1 programme/ 1 arbre)

2. Évaluation du fitness de chacun des individus de la population (Évaluation de l'adéquation des programmes au problème à résoudre)

3. Application des opérateurs de croisement, mutation, reproduction sur la population afin de créer une nouvelle population

4. Sélection des individus les mieux adaptés

5. Répéter les étapes 2 et 3 un certain nombre de fois

La PG comme instanciation des AGs

AG : population de solutions

PG : population de programmes

On retrouve les « ingrédients » des AGs :

problème de représentation
mesure de qualité (fitness)
pression de sélection
utilisation d 'une population
échanges d 'informations entre individus
…

Terminologie (1)

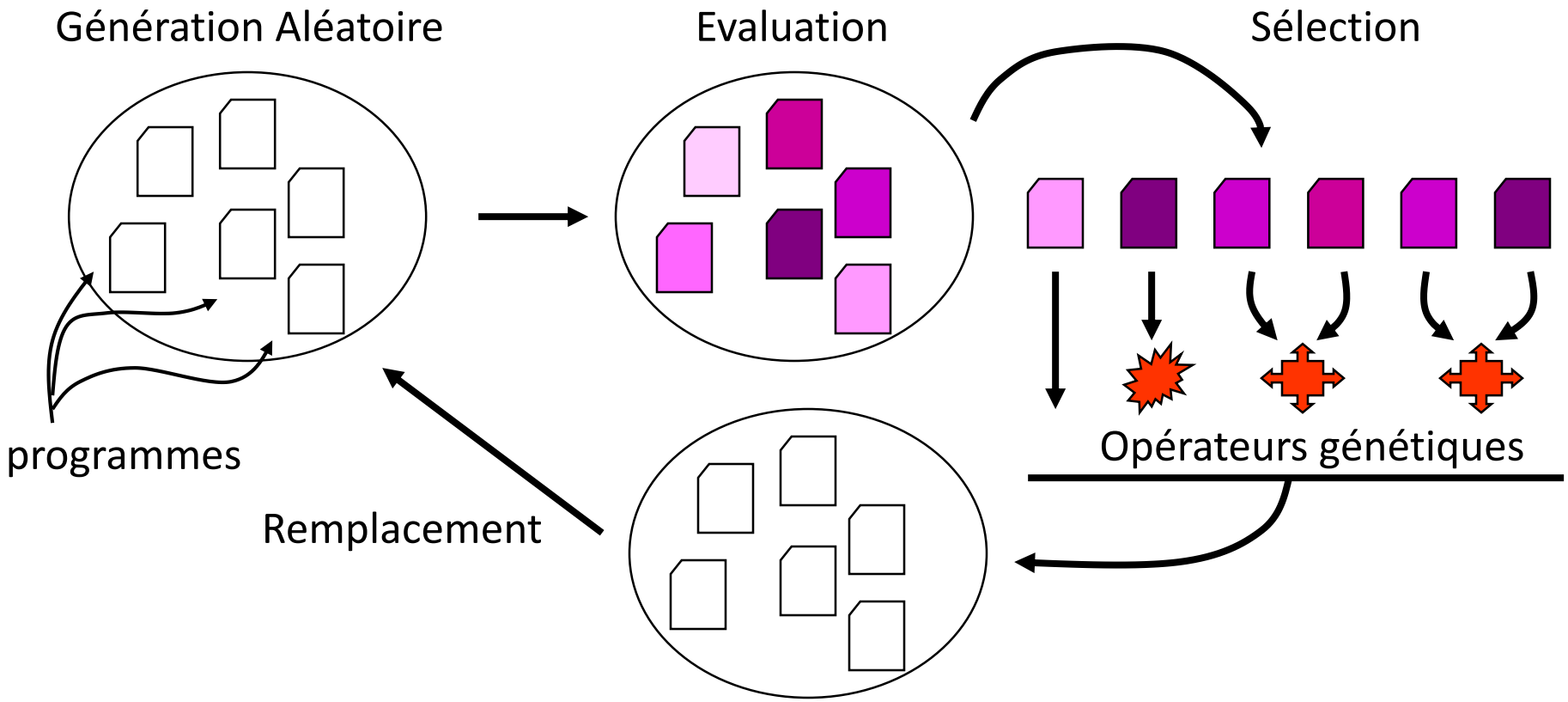
Les terminaux (feuilles de l 'arbre) :

- pseudo-variables contenant les entrées du programme;
- constantes, fixées d 'après la connaissance préliminaire du problème, ou générées aléatoirement;
- fonctions sans arguments mais avec effets de bord;
- variables ordinaires.

Terminologie (2)

- Les fonctions ou opérateurs (nœuds internes de l 'arbre) :
  - exemple : fonctions booléennes, arithmétiques, à effet de bord (assignation de variables, déplacement d 'un robot, ...), fonctions implantant des structures de contrôle : alternative, boucle, appel de routines, …
  - préférer un ensemble de fonctions petit et bien ajusté au domaine du problème, pour réduire l 'espace de recherche. Attention à ne pas le réduire trop, sous peine de perdre la possibilité de trouver des solutions intéressantes !

Schéma général de la PG



- La sélection est par le **fitness** des programmes
- Les opérateurs génétiques usuels sont la **copie**, la **mutation** et le **cross-over** bi-parental

Population initiale (1)

- On fixe une profondeur maximale pour les arbres.
- Création d 'arbres aléatoires par 2 méthodes principales :
  - « grow » : chaque nœud est tiré dans l 'ensemble {terminaux} + {fonctions}

  ⇨les arbres sont de forme irrégulière
  - « full » : on ne peut tirer un terminal que lorsque l 'on est à la profondeur maximum

  ⇨arbres équilibrés et « pleins »

Population initiale (2)

- Une synthèse, la méthode « ramped half & half » :
  - on va générer équitablement des arbres de profondeurs régulièrement échelonnées :

    2, 3, 4, …, maximum
  - à chaque profondeur, une moitie est générée par la méthode « full », l 'autre par la méthode « grow »

  ⇨L 'objectif est d 'obtenir plus de variabilité dans la population. C 'est la méthode préférentielle actuellement.

Evaluation, calcul du fitness (1)

- Comment évaluer la performance d 'un programme ? Tout dépend du problème.
- Quelques exemples possibles :

➢comparaison d 'images	nombre de pixels semblables
➢contrôle d 'un robot	nombre de chocs contre les murs
➢classification	nombre d 'exemples bien classés
➢vie artificielle	quantité moyenne de nourriture ingérée dans une simulation
➢regression de fonction	somme ou variance des erreurs sur un jeu d 'exemples

Evaluation, calcul du fitness (2)

- Exemple détaillé : régression symbolique / de fonction
  - On recherche une fonction à une entrée et une sortie satisfaisant le tableau suivant :

# cas	entrée: e_i	sortie: s_i
1	1	2
2	2	6
3	4	20
4	7	56
5	9	90

Chaque ligne représente un exemple d 'apprentissage ou « *fitness case* » : on dispose, pour chaque valeur en entrée, de la valeur attendue en sortie, que doit approximer au mieux le programme dont on calcule le fitness

Evaluation, calcul du fitness (3)

- Exemples de fonctions fitness usuelles :
  - somme des valeurs absolues des écarts entre valeur calculée par le programme et valeur attendue en sortie, pour chacun des *fitness cases* :

$$\varphi = \sum_{i=1}^{n} \left| P(e_i) - s_i \right|$$

– somme des carrés des écarts entre valeur calculée et valeur attendue (squared error) :

$$\varphi = \sum_{i=1}^{n} \left(P(e_i) - s_i \right)^2$$

Evaluation, calcul du fitness (4)

- variance de l 'erreur :

$$\varphi = \frac{1}{n}\sum_{i=1}^{n}\left(P(e_i) - s_i\right)^2$$

– écart-type de l 'erreur (root mean square error, erreur RMS) :

$$\varphi = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(P(e_i) - s_i\right)^2}$$

– écart-type relatif de l 'erreur (relative root mean square error, erreur RMS relative) :

$$\varphi = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{P(e_i) - s_i}{s_i}\right)^2}$$

Evaluation, calcul du fitness (5)

« *Une fonction fitness idéale devrait renvoyer une mesure différenciée et continue de l 'amélioration de la qualité des programmes* » *(Banzhaf et al.).*

Un peu de terminologie :

- « fitness standardisé » : la valeur du meilleur fitness possible est 0, toutes les valeurs de fitness sont positives. noté : f_s

- « fitness normalisé » : la valeur du fitness est toujours comprise entre 0 et 1.

- « fitness ajusté », un fitness normalisé où le meilleur score possible vaut 1 : $f_a = \frac{1}{1+f_s}$

La sélection

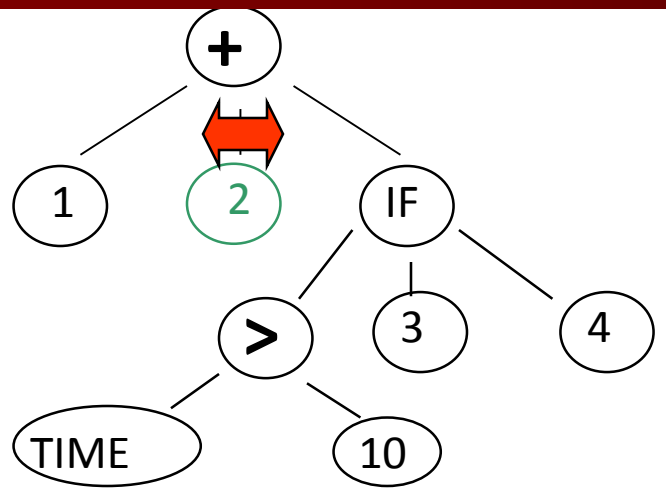
- On retrouve les méthodes de sélection utilisées dans les AGs :
  - sélection proportionnelle au fitness, avec éventuelle normalisation (scaling)
  - sélection basé sur le rang de l 'individu dans la population (ranking)
  - sélection par tournoi : la plus courante, car rapide et facilement parallélisable

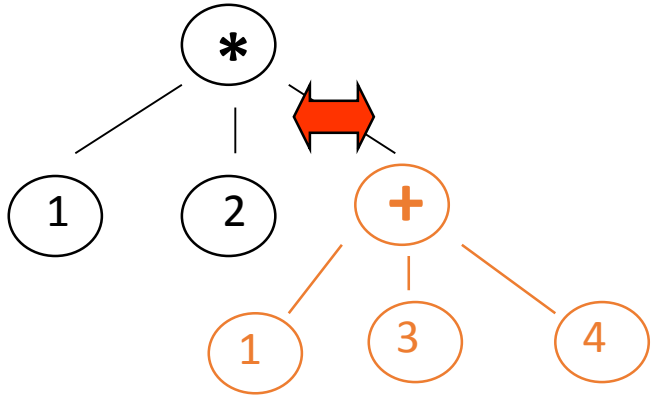
Opérateurs génétiques: copie

- Simple recopie d 'un individu d 'une génération à la suivante.

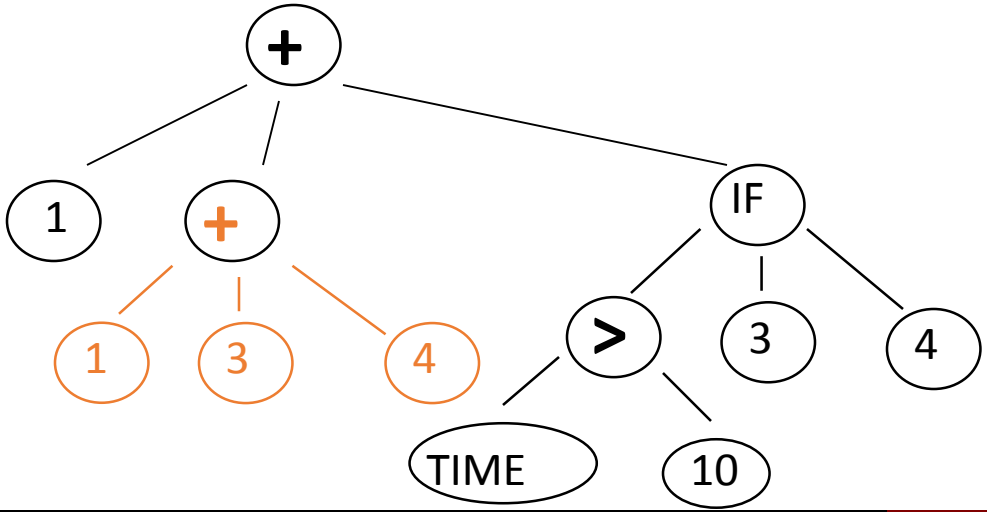
- Peut être forcée pour le meilleur individu : élitisme.

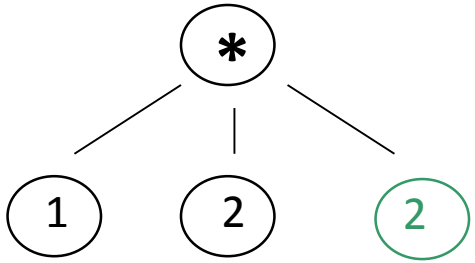
Opérateurs génétiques: crossover



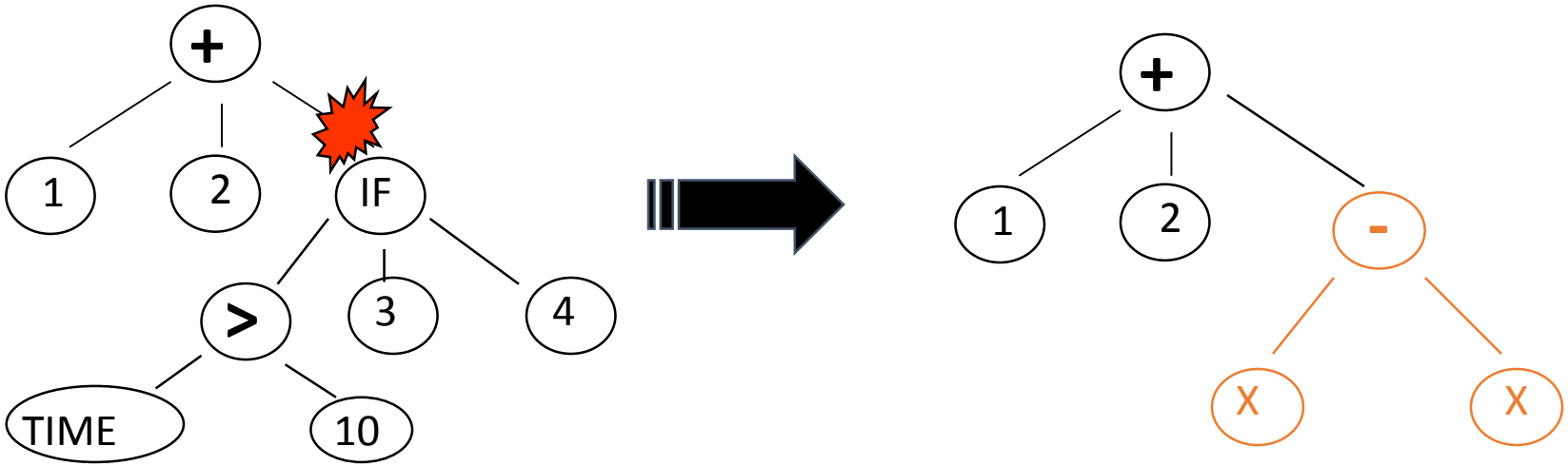


- Echange de deux sous-arbres pris aléatoirement





Opérateurs génétiques: mutation



- Destruction d 'un sous-arbre
- Remplacement par un sous-arbre aléatoire, créé comme lors de la génération de la population initiale.

Opérateurs génétiques

- Le cross-over ou la mutation sont susceptibles de transformer n 'importe quel sous-arbre argument d 'une fonction.

⇨Les fonctions doivent être capables d 'accepter toutes sortes de valeurs en argument, et il est préférable qu 'elle aient toutes le même type de valeur de retour (*propriété de clôture*)

⇨Exemple : remplacer la division standard par la division « protégée » qui renvoie 0 ou un grand entier en cas de division par 0.

Le remplacement

- Façon « AG » :
  - Générationnel : la nouvelle génération est souvent de même taille que l 'ancienne et elle la remplace.
  - « Steady state » : chaque nouvel individu est inséré dans la population au fur lors de sa création. Il remplace un individu déjà présent, par exemple celui de plus mauvais fitness, ou le moins bon du tournoi si sélection par tournoi.

Le remplacement

- Plusieurs tentatives de transposer le théorème des schémas des AGs vers la PG.
- Ces tentatives cherchent à modéliser l 'impact du cross-over sur des arbres.
- « Actuellement, aucune formulation du théorème des schémas ne prédit avec certitude la propagation des bons schémas pour la PG. » (Banzhaf *et al*. ou encore Angeline)
- Le problème vient notamment de la longueur variable des individus et du découplage important entre syntaxe et sémantique : la sémantique d 'un sous-arbre est assez indépendante de sa position dans l 'arbre solution.

Applications

- Tri (Kinnear), gestion de caches (Paterson et al.), compression de données (Nordin et al.), ...
- Reconnaissance d 'images (Robinson et al.), classification d'images (Zao), traitement d 'images satellitaires (Daïda), ...
- Prédiction de séries temporelles (Lee), génération d 'arbres de décisions (Koza), datamining (Raymer), …
- Classification de segments d 'ADN (Handley), de protéines (Koza et al.), ...
- Synthèse de circuits électroniques (Koza),
- Planification de déplacements de robot (Faglia et al.), évitement d 'obstacles (Reynolds) , mouvement de bras robotisés (Howley), …
- Modélisation en mécanique (Schoenauer et al.), …