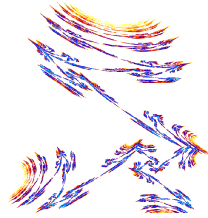


ALGORITHMES ÉVOLUTIONNAIRES

ET DARWINISME ARTIFICIEL



Evelyne LUTTON

Equipe COMPLEX - INRIA Rocquencourt - Evelyne.Lutton@inria.fr
<http://complex.inria.fr/>

COPIER LA BIOLOGIE DANS LES SYSTÈMES ARTIFICIELS

Utiliser des principes d'évolution organique en tant que technique d'optimisation globale.

Imiter les phénomènes d'apprentissage collectifs (adaptation) des populations naturelles.

Deux grands courants :

– algorithmes génétiques et systèmes de classeurs :

John Holland, David Goldberg.

– stratégies d'évolution et evolutionary computation :

Ingo Rechenberg, Hans-Paul Schwefel, Lawrence Fogel.

DARWINISME, ÉVOLUTIONNISME.

Charles Robert Darwin (1809-1882).

1831 - Voyage de 5 ans sur le HMS Beagle (îles Galapagos)

Ouvrage "Sur l'origine des espèces", Novembre 1859.

Selon Darwin, l'évolution se fonde sur 3 principes :

- importance des variations, macroscopiques et microscopiques au sein des espèces.
- lutte pour la vie,
- la sélection naturelle fait triompher la lignée qui possède la moindre variation utile à son environnement.

UN ENSEMBLE DE TECHNIQUES REGROUPÉES SOUS UN TERME GÉNÉRIQUE

Algorithmes Génétiques (AG),

Stratégies d'Évolution (SE),

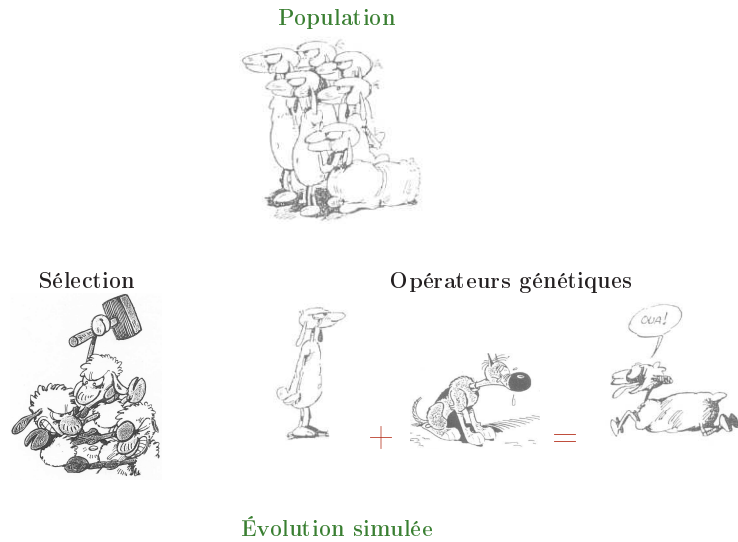
Programmation Évolutionnaire (PE),

Programmation Génétique (PG),

Systèmes de Classeurs (CS).

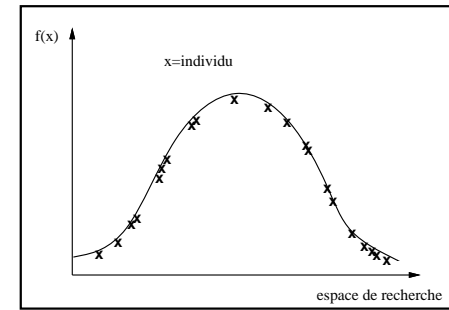
– Algorithmes Évolutionnaires (AE) =

INGRÉDIENTS DE BASE

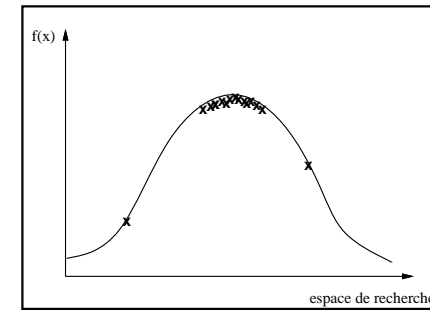


OPTIMISER UNE "ADAPTATION À L'ENVIRONNEMENT"

Solutions = individus d'une population.



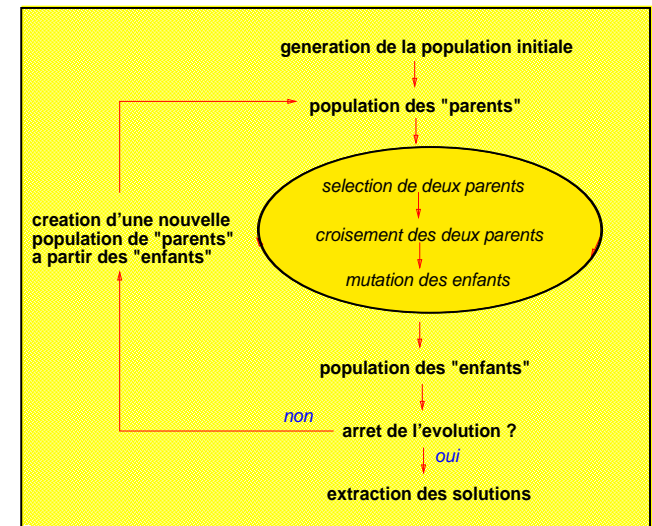
Population initiale aléatoire



Population évoluée

IMITER LA NATURE EN SIMPLIFIANT À L'EXTRÊME

Algorithme évolutionnaire	Méthode d'optimisation
individu	solution : vecteur
population	ensemble de solutions
chromosome	codage de la solution (binaire par exemple)
croisement ou recombinaison	opération sur deux codes
mutation	opération sur un code
environnement	espace de recherche
degré d'adaptation à l'environnement "fitness"	valeur de la fonction d'évaluation
évolution	maximisation de la fonction d'évaluation



LE MOTEUR ÉVOLUTIONNAIRE

Évaluation : estimer la qualité d'un individu.

→ "fitness", "performance", "fonction d'évaluation", "adaptation à l'environnement."

Sélection : trier les meilleurs individus.

→ tirages aléatoires biaisés (roulette wheel), sélection sur le rang, par tournoi.

Reproduction : appliquer les opérateurs génétiques, *croisements* et *mutations*, avec des probabilités p_c et p_m .

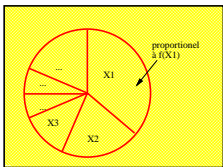
→ Lié à la représentation de l'espace de recherche :

Solutions ↔ chromosomes
Phénotypes ↔ génotypes

Remplacement : fabriquer la génération suivante.

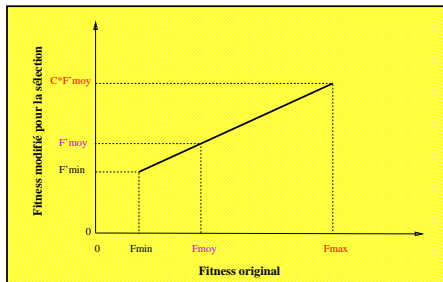
→ élitisme, pourcentage de renouvellement de la population.

LE RÔLE DE LA PRESSION SÉLECTIVE



$$P(x) = \frac{f(x)}{\sum_{x \in Population} f(x)}$$

→ **Scaling** : ni trop, ni trop peu.



INITIALISATION / ARRÊT DU PROCESSUS

Initialisation :

échantillonnage l'espace de recherche (aléatoire, régulier),
introduction de solutions initiales,
restrictions de la recherche.

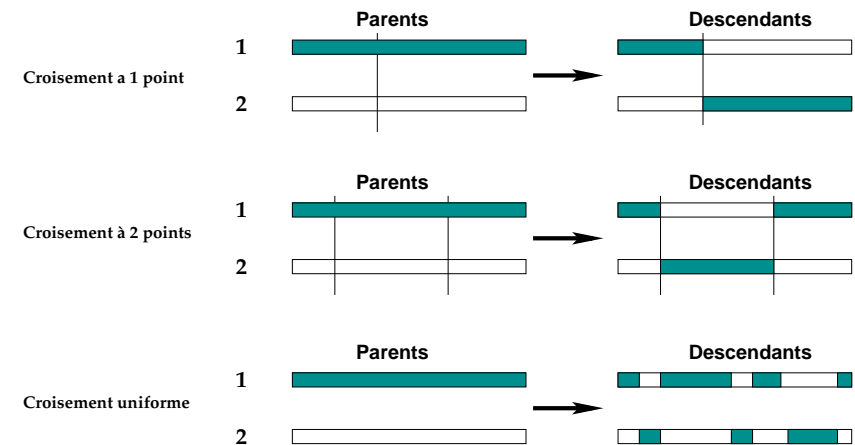
Arrêt, extraction des solutions :

le meilleur individu de la dernière génération !



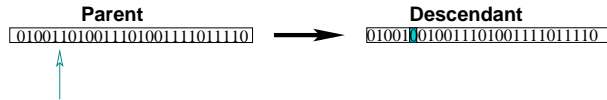
LA REPRÉSENTATION DISCRÈTE : ALGORITHMES GÉNÉTIQUES

Chaque individu est représenté par une chaîne (binaire) de longueur fixe.



Mutation

Une petite perturbation du génôme :



p_m est usuellement très faible et fixée.

Convergence prouvée si $p_m(k)$ décroît à chaque génération k en respectant :

$$p_m(k) \geq \frac{1}{2} * k^{-\frac{1}{M+L}}$$

M est la taille de la population et L est la longueur des chromosomes.

Gen : 0 Max : 21.000000 Min : 11.000000 Moy : 15.760000 Cmax : 2.0
 Gen : 1 Max : 23.000000 Min : 11.000000 Moy : 17.180000 Cmax : 1.9
 Gen : 2 Max : 22.000000 Min : 13.000000 Moy : 18.340000 Cmax : 1.6
 Gen : 3 Max : 24.000000 Min : 15.000000 Moy : 19.460000 Cmax : 2.0
 Gen : 4 Max : 25.000000 Min : 16.000000 Moy : 20.380000 Cmax : 2.0
 Gen : 5 Max : 26.000000 Min : 17.000000 Moy : 21.320000 Cmax : 2.0
 Gen : 6 Max : 27.000000 Min : 20.000000 Moy : 23.300000 Cmax : 2.0
 Gen : 7 Max : 28.000000 Min : 22.000000 Moy : 24.220000 Cmax : 2.0
 Gen : 8 Max : 28.000000 Min : 21.000000 Moy : 24.820000 Cmax : 1.8
 Gen : 9 Max : 29.000000 Min : 23.000000 Moy : 25.720000 Cmax : 2.0
 Gen : 10 Max : 30.000000 Min : 22.000000 Moy : 26.260000 Cmax : 1.8
 Gen : 11 Max : 30.000000 Min : 23.000000 Moy : 27.000000 Cmax : 1.7
 Gen : 12 Max : 32.000000 Min : 23.000000 Moy : 27.560000 Cmax : 1.9
 Gen : 13 Max : 32.000000 Min : 24.000000 Moy : 27.840000 Cmax : 2.0
 Gen : 14 Max : 32.000000 Min : 25.000000 Moy : 28.260000 Cmax : 2.0
 Gen : 15 Max : 32.000000 Min : 25.000000 Moy : 28.840000 Cmax : 1.8
 Gen : 16 Max : 32.000000 Min : 25.000000 Moy : 29.540000 Cmax : 1.5
 Gen : 17 Max : 32.000000 Min : 27.000000 Moy : 30.040000 Cmax : 1.6
 Gen : 18 Max : 32.000000 Min : 27.000000 Moy : 30.480000 Cmax : 1.4
 Gen : 19 Max : 32.000000 Min : 29.000000 Moy : 30.700000 Cmax : 1.7
 Gen : 20 Max : 32.000000 Min : 30.000000 Moy : 30.980000 Cmax : 2.0
 Gen : 21 Max : 32.000000 Min : 30.000000 Moy : 31.680000 Cmax : 1.1
 Gen : 22 Max : 32.000000 Min : 31.000000 Moy : 31.920000 Cmax : 1.0
 Gen : 23 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0
 Gen : 24 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0
 Gen : 25 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0
 Gen : 26 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0
 Gen : 27 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0
 Gen : 28 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0
 Gen : 29 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0
 Gen : 30 Max : 32.000000 Min : 32.000000 Moy : 32.000000 Cmax : 1.0

EXEMPLE : ONEMAX

- Chromosomes binaires, de longueur l_{max}

0101110010100

- fitness = nombre de "1" dans la chaîne

optimum : 111111111111

- population initiale : P chromosomes binaires aléatoires.

LA REPRÉSENTATION CONTINUE : STRATÉGIES D'ÉVOLUTION

La recherche se fait dans \mathbb{R}^n

Croisement barycentrique :

$$\forall i \in 1, \dots, n, \quad x'_i = \alpha x_i + (1 - \alpha) x_j$$

α , choisi par tirage uniforme dans $[0, 1]$ ou $[-\epsilon, \epsilon]$

Mutation Gaussienne :

$$\forall i \in 1, \dots, n, \quad x'_i = x_i + N(0, \sigma)$$

deux paramètres μ, σ

Mutation Log-normale auto-adaptative :

σ est intégré au code génétique :

$$\forall i \in 1, \dots, n, \quad \sigma'_i = \sigma_i \exp(N(0, 1))$$

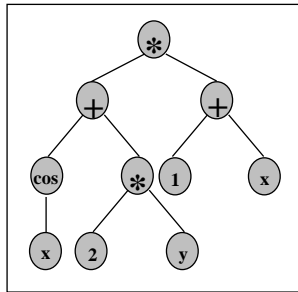
$$x'_i = x_i + N(0, \sigma'_i)$$

LA REPRÉSENTATION FONCTIONNELLE : PROGRAMMATION GÉNÉTIQUE

Créer des programmes sans programmer !!

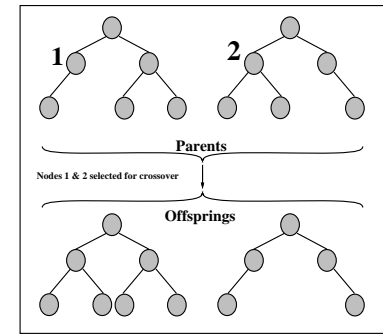
John Koza

- Structures de longueurs variables.
- Représentation par arbre.

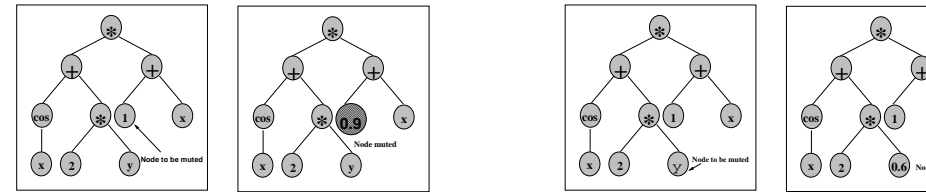


La fonction $((\cos(x) + 2 * y) * (1 + x))$ sous forme d'arbre.

Croisement



Mutations



LES SYSTÈMES DE CLASSEURS

CS = système d'apprentissage qui apprend des règles simples

1. Système de règles et de messages,

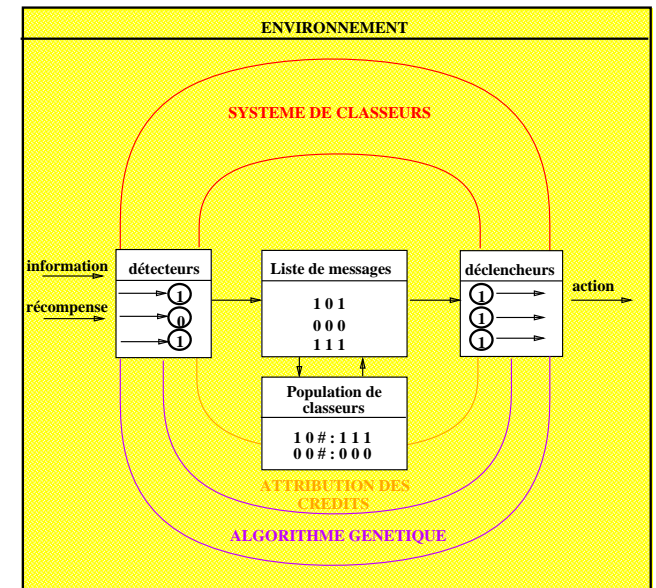
Message = chaîne binaire de longueur finie

Règle = classeur (*classifier*) $\langle \text{condition} \rangle : \langle \text{action} \rangle$

2. Système d'attribution des crédits,

Evaluation d'un classeur par un algorithme de "bucket-brigade".

3. Algorithme génétique : pour améliorer l'ensemble des classeurs.

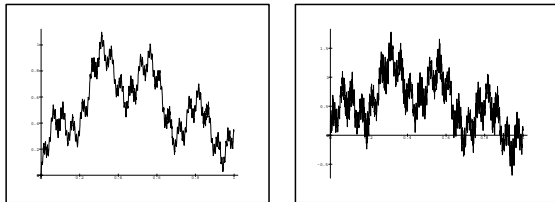


RÉSUMÉ

- Les AG fonctionnent sur une population de solutions, **codées**.
- La fonction à optimiser doit fournir une valeur positive au processus de sélection : **pas de continuité ni de dérivabilité**.
- La convergence se traduit par une **concentration de la population** autour de l'optimum global.
- La **vitesse de convergence** et la **qualité** de la solution finale dépendent :
du codage des chromosome,
de la forme de la fonction à optimiser,
des paramètres de l'AE.
- Les temps de calculs en version séquentielle peuvent être **longs**.

UTILISER UN AE

- cas **linéaire, convexe** : trop lent !
- cas **plus complexe** : exemple des fonctions fractales



Fonctions de Weierstrass de dimension 1.5 et 1.7

- cas **pseudo-aléatoire** : exemple d'une fonction en cryptographie

$$D_p\{x^4 \bmod p, a\} \text{ avec } a \text{ et } p \text{ des entiers codés sur 500 bits, } a < p$$

$$D_p\{x, y\} = \text{Min}\{(x - y) \bmod p, (y - x) \bmod p\}$$

ALGORITHMES GÉNÉTIQUES

(Goldberg)

- chromosomes = chaînes de caractères
- croisement à 1 site, à plusieurs sites, uniformes,
- stratégies de remplacement élitistes.

STRATÉGIES D'ÉVOLUTION

(Schwefel)

EVOLUTIONARY COMPUTATION

(Fogel)

- chromosomes $\in \mathbb{R}^n$,
- mutation Gaussienne, croisements barycentriques,
- stratégies de remplacement (μ, λ) , (μ, μ)

GENETIC PROGRAMMING

(Koza)

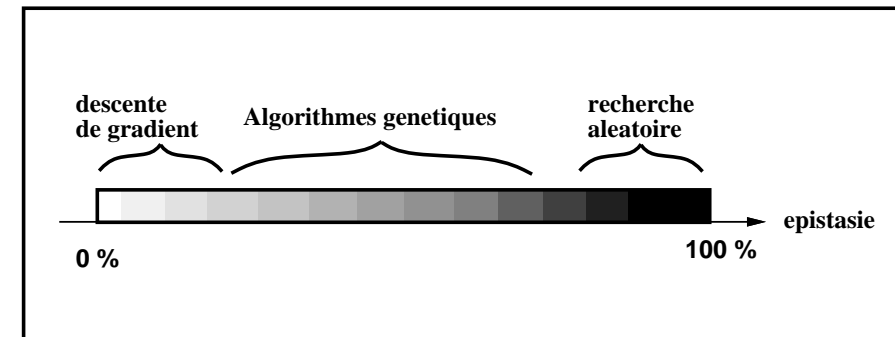
- chromosomes de taille variable (arbre)

CLASSIFIER SYSTEMS

(Holland, Goldberg)

- chromosomes = règles d'un système expert

Recherche d'un optimum global dans un environnement complexe et multidimensionnel

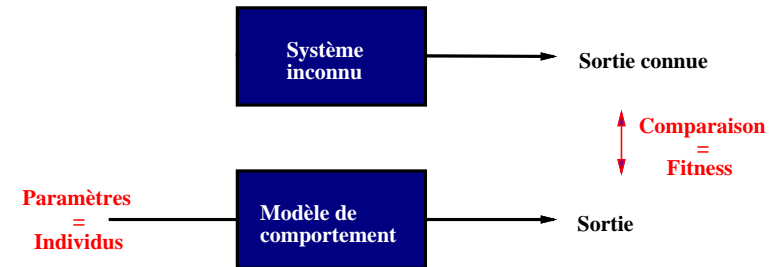


Adaptation à un environnement variable → IA, contrôle, commande de processus

CONSTRUIRE DES APPLICATIONS

- Identifier l'espace de recherche et les **contraintes** du problème.
- Choisir un codage **efficace** (gérer la redondance).
- Construire une fonction de fitness **économique**.
- Bien régler les **opérateurs, stratégies, paramètres**.

RÉSOLUTION DE PROBLÈMES INVERSES PAR APPROCHE EN "BOÎTE NOIRE"



ALGORITHMES ÉVOLUTIONNAIRES NON STANDARDS

Niches écologiques : détection de plusieurs optima.

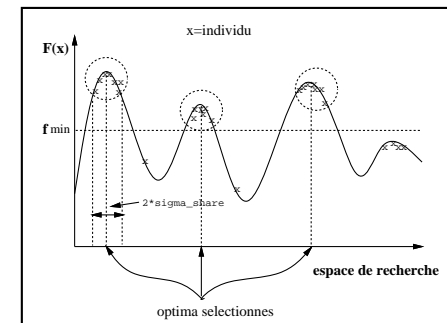
Optimisation multi-objectif : échantillonnage du front de Pareto.

Coévolution-coopération : approche Parisienne.

Evolution interactive.

NICHAGE ET ENVIRONNEMENT MULTIMODAL

Coloniser plusieurs optima



- par maintien de la diversité génétique
(Caviccio, De Jong, Mauldin),

- par partage des ressources
(Goldberg, Richardson),

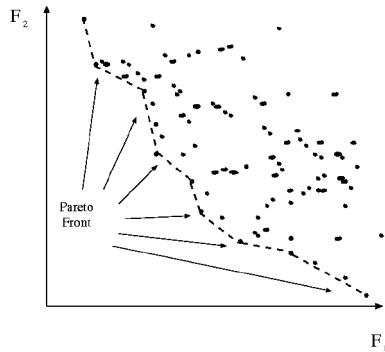
$$\text{Fitness}'(x) = \frac{\text{Fitness}(x)}{\sum_{x' \in \text{Vois}(x)} \text{Sh}(d(x, x'))}$$

$$\text{Sh}(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{\text{share}}}\right)^\alpha & \text{si } d < \sigma_{\text{share}} \\ 0 & \text{sinon} \end{cases}$$

- par restriction des croisements.

OPTIMISATION MULTICRITÈRE

- **Front de Pareto** \equiv Meilleur compromis
- sélection spécifique + nichage
- En une fois, un EA produit un échantillonnage du front de Pareto.



Front de Pareto = ensemble des solutions non dominées

ALGORITHMES ÉVOLUTIONNAIRES PARISIENS

Principe : La solution est représentée par plusieurs individus dans la population.

- Ce n'est plus exactement une optimisation.
- Plus délicat à mettre en œuvre.
- Gain de temps d'exécution important.
- Meilleure utilisation des informations a priori.

Actuellement :

- Résolution du problème inverse pour les IFS.
- Art et design.
- Text retrieval.
- Vision stéréo.

ALGORITHMES ÉVOLUTIONNAIRES INTERACTIFS

Principe : l'humain dans la boucle évolutionnaire.

- On interroge l'utilisateur régulièrement (fitness interne/fitness externe).
- Il peut faire des "optimisations locales", contrôler le processus à plusieurs niveaux.
- Permet d'optimiser des paramètres "subjectifs" difficiles à modéliser autrement.

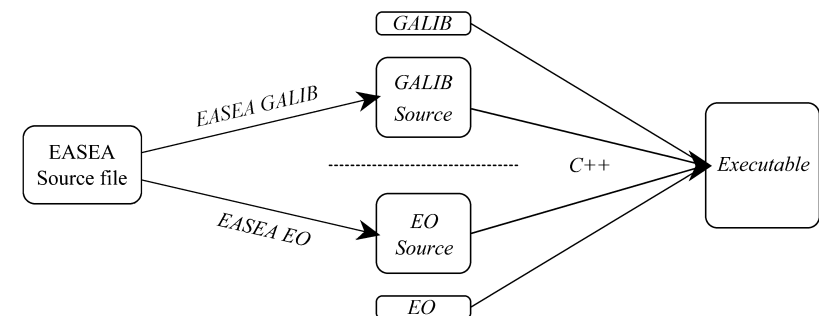
Actuellement :

- art et design (ArtiE-Fract et l'atelier des fractales),
- text retrieval (Novartis Pharma),
- e-learning (Paraschool).

EASEA : EAasy Specification for Evolutionary Algorithms

Langage de spécification d'Algorithmes Évolutionnaires.

www-rocq.inria.fr/



```
/****** onemax.ez *****/
```

```
\User declarations :
```

```
#define SIZE 10  
float pMutPerGene=0.1;
```

```
inline void swap(bool& a, bool& b)  
{bool c=a; a=b; b=c;}  
\end
```

```
\User classes :
```

```
GenomeClass { bool x[SIZE]; }  
\end
```

```
\GenomeClass::initialiser : // "initializer" is also accepted  
for (int i=0;i<SIZE;i++) Genome.x[i]=tossCoin(.5)?1:0;  
\end
```

```
\GenomeClass::crossover :  
int CrossoverPosition=random(0,SIZE);
```

```
for(int i=0;i<CrossoverPosition+1;i++)  
swap(child1.x[i],child2.x[i]);  
\end
```

```
\GenomeClass::mutator : // Must return the number of mutations
```

```
int NbMut=0;  
for (int i=0;i<SIZE;i++)  
if (tossCoin(pMutPerGene)){  
NbMut++;  
Genome.x[i]=Genome.x[i]?0:1;  
}  
return NbMut;  
\end
```

```
\GenomeClass::evaluator : // Returns the score
```

```
int Score=0;  
for (int i=0; i<SIZE;i++)  
Score+=(int)Genome.x[i];  
return Score;  
\end
```

```
\Default run parameters : // Please let the parameters appear  
// in this order
```

```
Number of generations : 15 // NB_GEN  
Mutation probability : 1 // MUT_PROB  
Crossover probability : 1 // XOVER_PROB  
Population size : 30 // POP_SIZE
```

```
Offspring population size : 12 // 40%  
Replacement strategy : Plus // Comma, SteadyState, Generational  
Discarding operator : Worst // Best, Tournament, Parent, Random  
Evaluator goal : Maximise // Minimise  
Elitism : On // Off  
\end
```