Introduction aux Algorithmes Génétiques (extraits)

G.MENIER - Maître de Conférences

Apprentissage / DEA-DESS / Université de Bretagne Sud

gildas.menier@univ-ubs.fr
Institut Universitaire Professionnalisé
Informatique & Statistiques
rue Yves Mainguy, 56000 Vannes

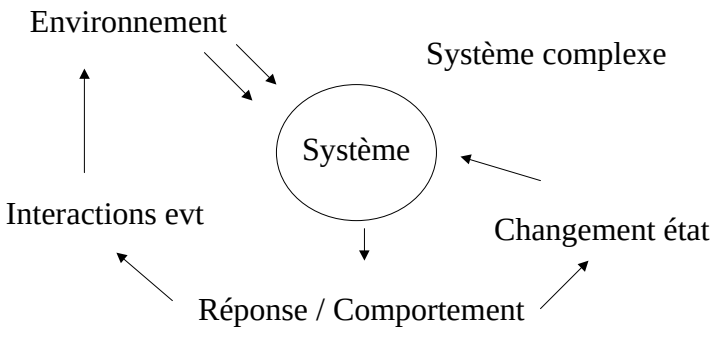
Algorithmes Génétiques

- Introduction
  - Systémique
  - Apprentissage et optimisation
  - Principales méthodes d ’optimisation
- Méthodes évolutionnistes
- Algorithmes génétiques
  - Introduction et principes
  - Un exemple simple
  - Théorème des schemata
  - Applications

gildas.menier@univ-ubs.fr - Algorithmes Génétiques et apprentissage

Introduction

Introduction : Systémique et apprentissage



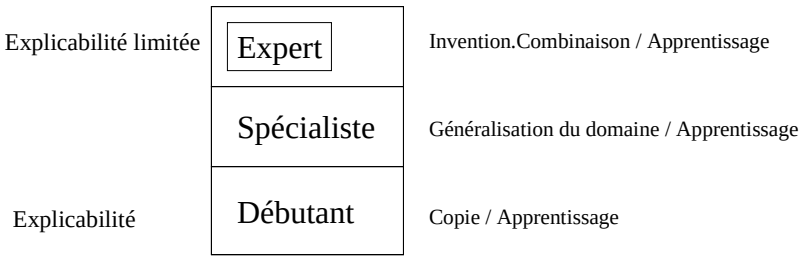
Homéostasie / Autopoièse

Introduction : Systémique et apprentissage

- Apprentissage
  - Changement d ’état ?
  - Modification du comportement ?
  - Influence de l’environnement ?
  - Notion de temps ?
- Système
  - Ensemble de paramètres qui interagissent de manière plus ou moins accessible à un observateur et définissent un comportement

Introduction : Apprentissage et Optimisation

- Apprentissage
  - Modifier le comportement
  - Modifier les paramètres du système

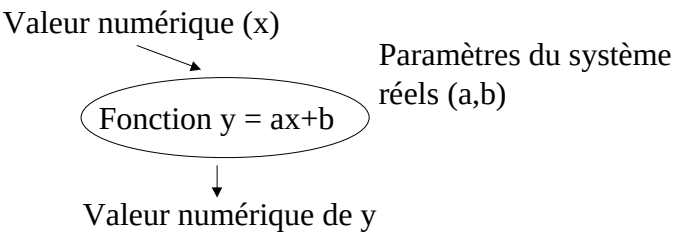


Introduction : Apprentissage et Optimisation

- Optimisation
  - Définition du comportement à atteindre
  - Calcul des paramètres du système pour atteindre
- Solution
  - Un ensemble de paramètres du système permettant d 'obtenir (+/-) le comportement souhaité

Introduction : Apprentissage et Optimisation

- Exemple :



Comportement recherché : donne 5 pour 3 en entrée

Ensemble d 'apprentissage

Introduction : Apprentissage et Optimisation

- Résolution
  - Analytique
- Solutions : une infinité
  - Exactes : (0,5); (2, -1) etc…
  - Approchées : (0,5.1); (0,4.9); (1,4.6) etc…
- Solutions approchées
  - Critère pour ‘ approchées ’
  - Lesquelles sont les meilleures ?

Introduction : Apprentissage et Optimisation

- Évaluation d’une solution
  - Pour un ensemble de paramètres (solution approchée), évaluer la différence entre le comportement approché et le comportement souhaité
  - On cherche à rendre meilleur le comportement approché en utilisant cette mesure
  - On cherche à minimiser la distance entre le comportement souhaité et le comportement approché
  - Optimisation de cette mesure = apprentissage

Introduction : Apprentissage et Optimisation

- Formalisme :

  Ensemble de paramètres du système : P_{set} : P_0, … P_n

  (n peut faire partie des paramètres du système)

  Comportement : $C(P_{set})$ → stratégies

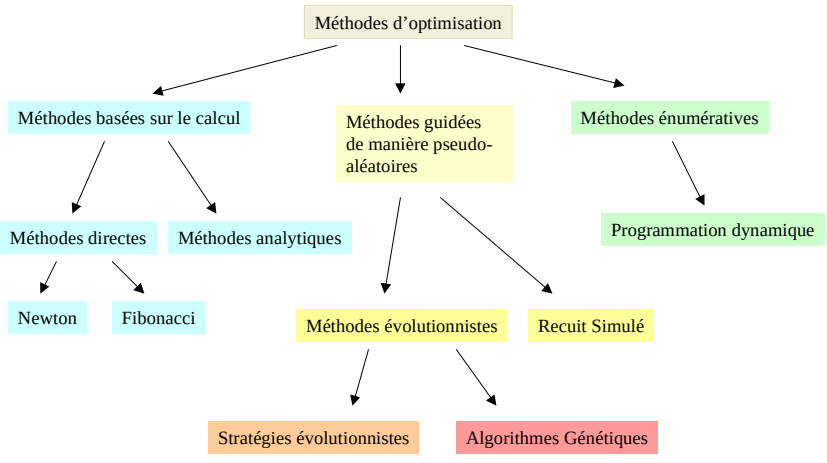
  Comportement souhaité : C_{target}

  Mesure $F_{distance}(C(P_{set}, C_{target}))$

  Valeur de la solution $S_{Ctarget}(P_{set})$ = Transfert($F_{distance}(C(P_{set}, C_{target}))$)

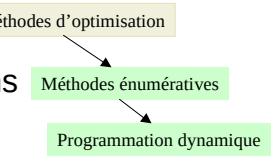
- Objectif :
  - maximiser la valeur d 'une solution
  - minimiser la différences des comportements

Introduction : Méthodes d'optimisation



Introduction : Méthodes d'optimisation

- Recherche énumérative
  - évaluation de toutes les solutions
  - on conserve la meilleure



MAIS – Parcours de l 'espace paramétrique

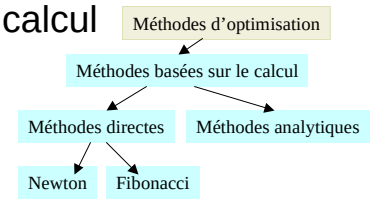
- Programmation dynamique
  - Heuristiques de recherche
  - Algorithmes A*, Alpha/Beta

MAIS – Attention : évaluation de la complexité AVANT

- Exemples : Echecs, Logistique, passwd

Introduction : Méthodes d'optimisation

- Méthodes basées sur le calcul
  - Résolution mathématique



MAIS – Connaissance

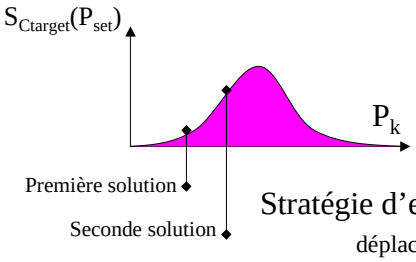
- Méthodes analytiques
  - Fonction $S_{Ctarget}(P_{set})$ connue

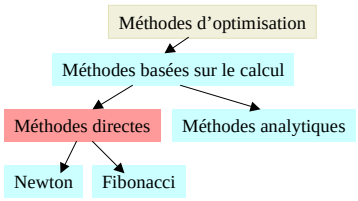
MAIS – résolution directe si possible

- Méthodes directes
  - Méthodes de recherche par voisinage
  - Méthodes gradient

Introduction : Méthodes d'optimisation

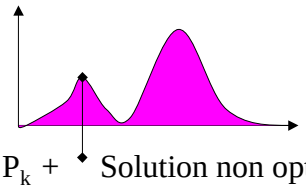
- Méthodes directes





Stratégie d'exploration locale :
déplacement dans le sens de gradient +

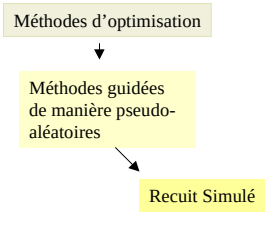
MAIS



Gestion des P_k + Solution non optimale - blocage

Introduction : Méthodes d'optimisation

- Méthodes guidées de manière pseudo-aléatoire
  - Méthode directe +
  - éviter pb recherche locale



- Recherche Aléatoire
  - Génération aléatoire de solutions
  - Vecteurs dans l 'espace des paramètres

MAIS
  - Coup de bol
  - Long ?
  - Terminaison ? (cf recherche par énumération)

Introduction : Méthodes d'optimisation

- Recuit Simulé
  - Kirkpatrick, Gelatt, Vecchi (83)
  - analogie thermodynamique / crystalisation
  - état d'un gaz = état des particules
  - Probabilité de trouver le système dans un état donné prop au facteur de Boltzmann $e^{-H(e)/T}$
    - H(e) énergie de cet état
    - T température
  - Probabilité d'occurrence de deux états e_1 et e_2:

$$\frac{p(e_1)}{p(e_2)} = e^{(\frac{-(H(e_1)-H(e_2))}{T})}$$

  - Metropolis (53)
    - Simulation d 'un ensemble d'atomes à l 'équilibre à T
    - évaluation de la variation d 'énergie ΔH

Introduction : Méthodes d'optimisation

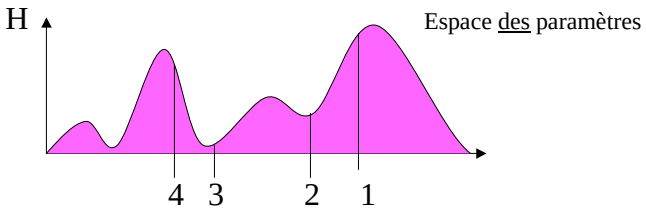
- Recuit Simulé:
  - Principe de Metropolis
    - A chaque étape de l'algo, on envisage le déplacement aléatoire d 'un atome (changement de l 'état du système)
    - ΔH est évalué
    - Si ΔH < 0, le déplacement est conservé (état + stable)
    - sinon, on conserve le déplacement avec une probabilité :

$$e^{-\Delta H(e)/T}$$

    - On fait décroître la température T : on diminue la probabilité d 'accepter un état qui augmente l'énergie
  - Le système atteint une valeur minimale pour H

Introduction : Méthodes d'optimisation

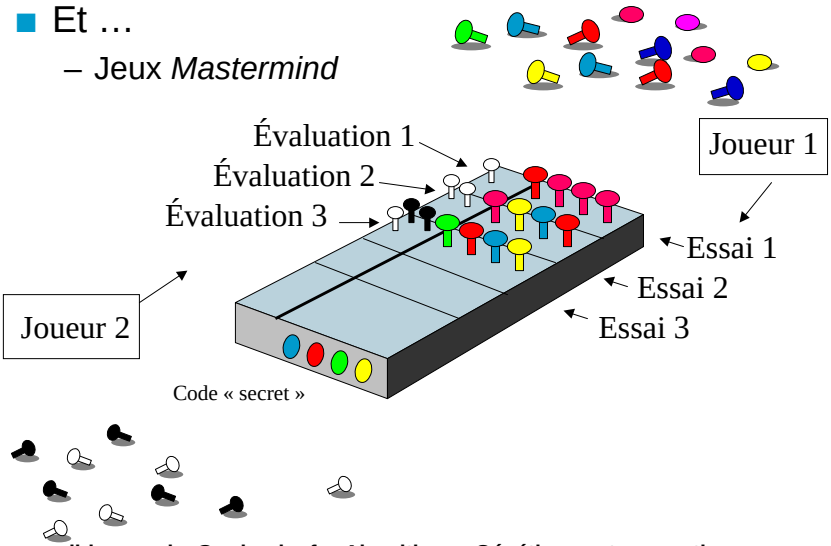
- Recuit Simulé:
  - Principe :
    - On ne considère qu'un point de l'espace
    - On remplace H (énergie) par une fonction à minimiser
    - Simulation (décroissance T, terminaison etc..)



    - Analogie des ' billes '
    - Permet d ' #éviter# les minima/maxima locaux (gradient)

Introduction : Méthodes d'optimisation

- Et …
  - Jeux *Mastermind*



Introduction : Méthodes d'optimisation

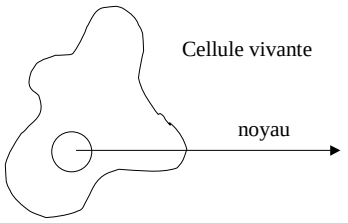
Théorie de l'invention

?

Algorithmes Génétiques

Algorithmes Génétiques

- 1. Introduction et principes



Comportement global régit par informations génétiques
‘ programmation ’ du vivant

Interprétation d ’un ‘ programme ’ matériel génétique / chromosomes
= **génotype**

Nature / Formes / capacité etc… = conséquences de l ’interprétation du matériel génétique
= **phénotype**

La reproduction est conditionnée par l ’adaptation au milieu naturel
L ’évolution = adaptation du matériel génétique pour maximiser la probabilité de reproduction : optimiser les réactions au milieu naturel

Par rapport au matériel génétique : Evolution = mécanisme d ’apprentissage

Algorithmes Génétiques

- 1. Introduction et principes

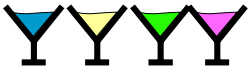
Algorithme génétique = Simulation Mécanismes évolutionnistes

Biologie cellulaire	Algorithme Génétique
Chromosomes	Chaîne
Gènes	Trait/caractéristique
Allèle	Valeur d ’une caractéristique
Locus	Position dans la chaîne
Génotype	Structure (codage)
Phénotype	Informations issues du décodage
Reproduction	Reproduction
Croisement	Croisement
Mutation	Mutation
Espèce	Niche écologique

Algorithmes Génétiques

- 2. Exemple Simple : idées générales

  - Barman : composition du cocktail idéal
  - Pas de fonction connue (le client évalue)
  - Recherches en // sur n verres
  - Chaque verre représente une solution approchée au problème d 'optimisation

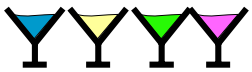


  - Les n solutions approchées constituent une *population* génétique

Algorithmes Génétiques

- Exemple Simple : idées générales

  - La première population est tirée au hasard ou bien confectionnée en fonction de l'expérience du Barman
  - Le Barman fait goûter chaque solution et demande au client de la noter
  - La note que reçoit un individu de la population est l '*adéquation* génétique (valeur d 'adaptation) de cet individu



Valeur d'adéquation 1 3 5 1

  - Le Barman va produire une nouvelle population de cocktail en fabriquant de nouveaux cocktails à partir des meilleurs
  - On parle de nouvelle *génération*

Algorithmes Génétiques

- Exemple Simple : idées générales
  - *Croisement* : Il peut mélanger deux cocktails pour obtenir un des nouveaux individus de la population
  - *Mutation* : Il peut éventuellement rajouter un ingrédient (au hasard) de temps à autre ou de changer un des ingrédients.
  - *Sélection* : Il choisit d 'utiliser en priorité les anciens bon cocktails pour ses nouveaux mélanges et de balancer les mauvais cocktails de la *génération* précédente.
  - Ces trois opérations sont appelées *opérateurs génétiques*

Algorithmes Génétiques

- Exemple Simple : idées générales
  - Le Barman met en œuvre un processus de sélection évolutionniste qui répond à une contrainte de l 'environnement (ici représenté par le client).
  - Le mécanisme de sélection favorise la survie des meilleurs cocktails et donc la recherche de la recette idéale.
  - Quand le Barman décide d 'arrêter, il peut choisir de ne conserver que le meilleur cocktail, ou les meilleurs cocktails de la population.
  - A chaque génération, on ne conserve que n cocktails (les générations précédentes sont ' perdues ').

Algorithmes Génétiques

- Exemple Simple : implémentation
  - Définition des paramètres pertinents (ex : un paramètre)
  - Codage Génotype : chaîne génétique

    0 0 1 0 1 1 0 0
  - Codage Phénotype : traduction numérique/symbolique

    0 0 1 0 1 1 0 0 → 46
  - Population de solutions : ensemble de chaînes génétiques*
  - Fonction d 'adéquation : évaluation de la pertinence d 'une solution (génotype+fitness = phénotype)
  - Fonction ' cachée ' (pas d 'informations *a priori*)

Algorithmes Génétiques

- Exemple Simple : Population initiale
  - Population de départ (4 chaînes)

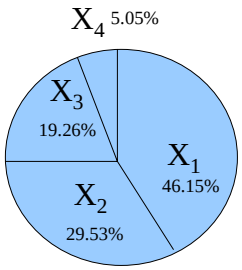
	Codage	Valeur	Adéquation
X_1	10000010	130	16900
X_2	01101000	104	10816
X_3	01010100	84	7056
X_4	00101011	43	1849

  - Aléatoires ou pas (heuristiques)

Algorithmes Génétiques

- Exemple Simple : Reproduction

  – 1. Opérateur de Sélection / Reproduction



‘ Tirage de la roue de loterie ’

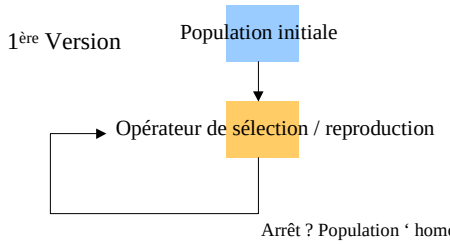
Codage	Valeur	Adéquation	Contribution
10000010	130	16900	46.15%
01101000	104	10816	29.53%
01010100	84	7056	19.26%
00101011	43	1849	5.05%
		36621	100%

Somme

Tirage aléatoire d ’une nouvelle population avec probabilités relatives aux contributions

Algorithmes Génétiques

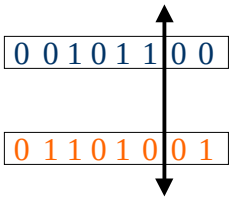
- Exemple Simple : Algorithme



La somme des adéquations augmente
La population est statistiquement envahie par le meilleur individu (chaîne qui a la meilleure valeur d ’adéquation)
= Tirage aléatoire + conservation du meilleur

Algorithmes Génétiques

- Exemple Simple : Croisement
  - 2. Opérateur de croisement
    - mise en place d 'un mécanisme de communication entre solutions



Position de croisement hasard
création d 'une (2) nouvelles chaînes
par échange d'information génotype
Crossover

On peut décider de conserver
une chaîne ou les deux (conser-
vation de l 'information totale)

0 1 1 0 1 0 0 0

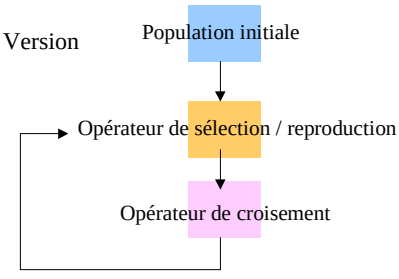
0 0 1 0 1 1 0 1

Par exemple

Algorithmes Génétiques

- Exemple Simple : Algorithme

2nd Version



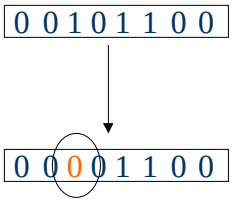
Probabilité de choix des chaînes à
croiser peut dépendre de leur adéquation
(*Fitness*)
Limitation : la diversité des chaînes
possible dépend de la population
initiale.
Mécanisme semblable à un automate de
recherche aléatoire à états finis

Condition d 'arrêt ?

Délicat : difficile de faire des hypothèses sur l 'évolution
de la population
difficile d 'estimer la valeur maximale de l 'adéquation
etc...

Algorithmes Génétiques

- Exemple Simple : Mutation

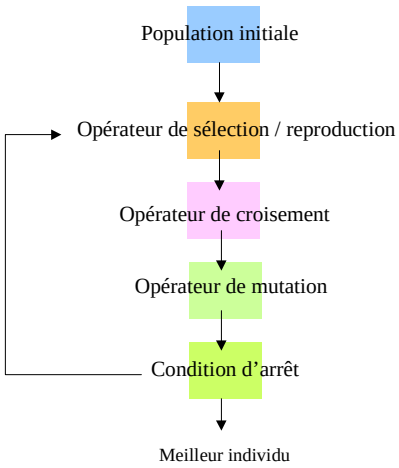


Lors de la reproduction, probabilité ‘ faible ’ d ’altérer le génotype
exemple : inversion d ’un bit

Objectif : apporter une diversité en cas de ‘ stagnation ’ de la population des solutions

Algorithmes Génétiques

- Principe de l ’algorithme génétique de base

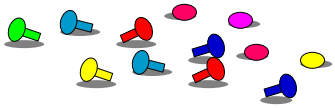


Paramètres :
- Taille population ?
- Codage ?
- Générateur aléatoire ?
- Fonction d’adéquation ?
- Stratégie de reproduction ?
- Stratégie de croisement ?
- Stratégie de mutation ?
- Probabilité de mutation ?
- Probabilité de croisement ?
- Condition d’arrêt ?
- etc...

Mise en Œuvre difficile

Introduction : Méthodes d'optimisation

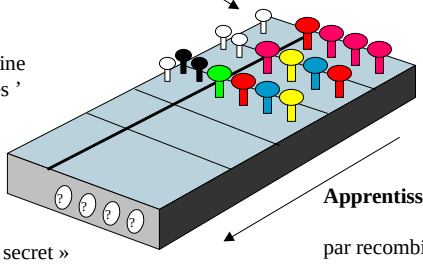
- *MasterMind ?*



Évaluation de la chaîne
Fitness
Adéquation
Adaptation

Apprentissage

par sélection : on élimine
les chaînes ' mauvaises '



Chaîne génétique

Apprentissage

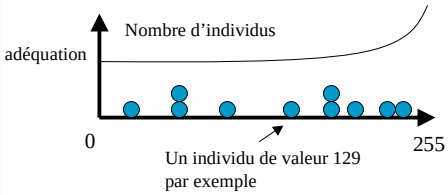
par recombinaison des anciennes
solution ' intéressantes '
(croisement)

par introduction de nouvelles
infos (mutation)

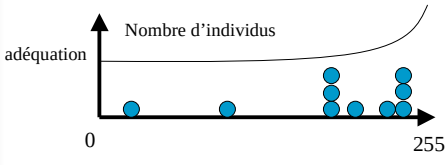
Code « secret »
Objectif (C_{target}) / cible

Algorithmes Génétiques

- Exemple d 'exécution : population de n=10 individus



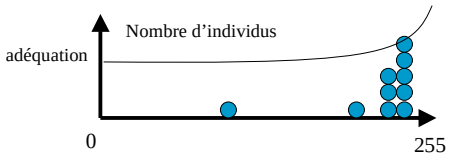
Population de départ
Tirage aléatoire de n=10 chaînes
Loi uniforme



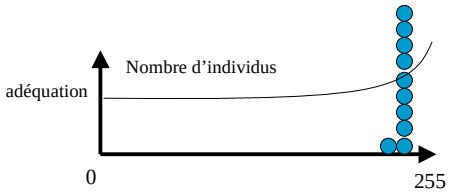
Fin de la reproduction 1
on conserve en moyenne les
meilleurs individus

Algorithmes Génétiques

- Exemple d 'exécution : population de n=10 individus



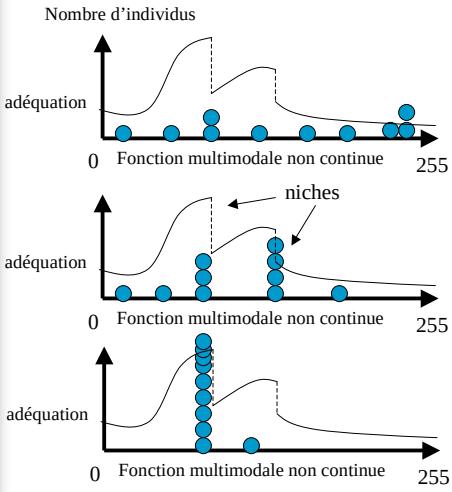
Fin de la reproduction 3
L 'ensemble de la population ' niche '
vers les zones d 'adéquation optimale



Fin de la reproduction ' k '
L 'ensemble de la population ' niche '
vers les zones d 'adéquation optimale

Algorithmes Génétiques

- Exemple d 'exécution : population de n=10 individus



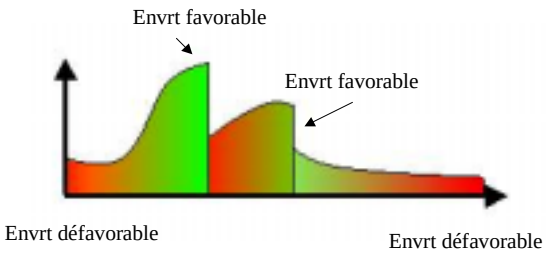
Dans le cas de fonctions d 'adéquation non dérivable, non continue, non intégrable de manière analytique (éventuellement non définie sur tout l 'espace de recherche), l 'algorithme génétique tend à répartir sa population dans les régions les plus profitables à la survie de la population des solutions.

La solution éventuellement optimale finale permet de trouver le maximum globale en évitant les extrema locaux (gradient).

L 'analyse de recherche sous-optimale permet de détecter des régions intéressantes ces régions sont appelées de ' niches écologiques '

Algorithmes Génétiques

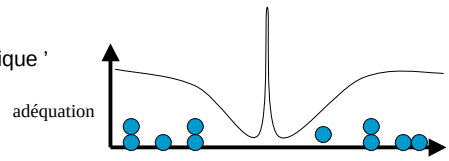
- Interprétation évolutionniste



Les contraintes d 'envr dont représentées par la fonction d 'adéquation.
Les individus doivent adapter leur matériel ' génétique ' pour permettre leur survie / survie de la population pour 'apprendre' à réagir de manière optimale face à l 'environnement.
On peut considérer un algorithme génétique comme un mécanisme d 'apprentissage par contrainte (sélection/reproduction) - pénalisation.

Algorithmes Génétiques

- Remarques :
  - Cas du ' poteau télégraphique '

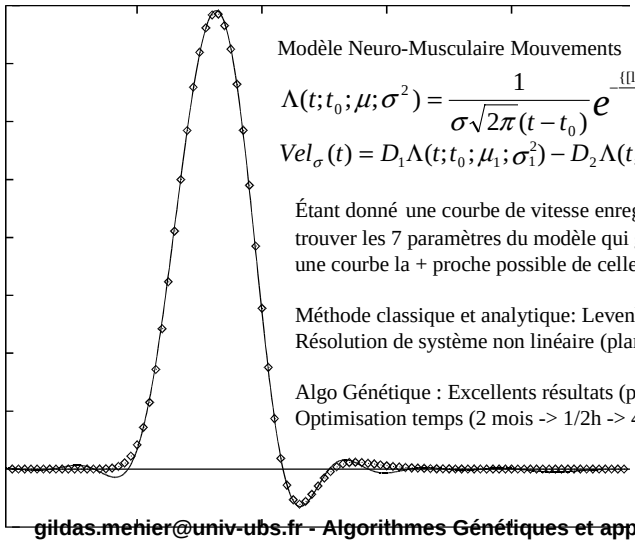


Dans le cas où ce problème n 'admet pas de solutions analytiques les algorithmes génétiques donnent quand même les meilleurs résultats (toutes méthodes confondues)

  - Recuit simulé et Algorithme Génétique
    - Recuit = une région explorée à la fois + déplacement stochastique
    - AlgGen = une population / n régions maxi explorées en // (avec communication entre les solutions via le crossover)
    - *Genetic Annealing* = Recuit Génétique : contrôler l 'opérateur de mutation à la manière du recuit : Excellents résultats pour des fonctions numériques
    - Recuit / Algo Génétiques complémentaires
  - La fonction d 'adéquation peut être multidimensionnelle
    - Dimension = nombre de paramètres d 'exploration

Algorithmes Génétiques

- Exemple d 'application multidimensionnelle (Ménier/Plamondon)



Modèle Neuro-Musculaire Mouvements

$$\Lambda(t;t_0;\mu;\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}(t-t_0)} e^{-\frac{\{[\ln(t-t_0)-\mu]^2\}}{2\sigma^2}}$$

$$Vel_\sigma(t) = D_1\Lambda(t;t_0;\mu_1;\sigma_1^2) - D_2\Lambda(t;t_0;\mu_2;\sigma_2^2)$$

Étant donné une courbe de vitesse enregistrée, trouver les 7 paramètres du modèle qui génèrent une courbe la + proche possible de celle observée.

Méthode classique et analytique: Levenberg-Marquardt
Résolution de système non linéaire (plante)

Algo Génétique : Excellents résultats (pas de plantage)
Optimisation temps (2 mois -> 1/2h -> 40s, PII400Mhz)

Algorithmes Génétiques

- 3. Fonctionnement des Algo Génétiques
  - Interprétation par Chaînes de Markov (biblio)
  - Interprétation par énumération spatiale et transformée de Walsh (biblio)
  - Théorème des *Schemata (Holland)*

Algorithmes Génétiques

- Théorème des Schemata
  - Hypothèse : les chaînes sont construites sur un alphabet binaire
  - Alphabet $A_b = \{0,1\}$
  - Chaîne génétique : C = $b_1b_2b_3b_4b_5b_6b_7b_8$ (par exemple)
  - ' *L* ' représente la longueur d 'une chaîne
  - Une génération : *G(t)* génération au temps *t*.
  - Une configuration de bits particulière = un *schema* (pl.*Schemata*)
  - Un schema est défini sur : $A_t = \{0,1,*\}$
  - * représente un 0 ou un 1
  - Par exemple, le schema S = 00*010*1 représente l 'ensemble :
    - {00<u>0</u>010<u>0</u>1, 00<u>1</u>010<u>0</u>1, 00<u>1</u>010<u>1</u>1, 00<u>0</u>010<u>1</u>1}
  - La chaine C = 00101001 est un *représentant* du schema S
  - S_g = ******** est l 'ensemble des valeurs possible
  - S est plus spécifique que S_g

Algorithmes Génétiques

- Théorème des Schemata
  - L 'ordre d 'un *schema* est le nombre de positions fixées 0 ou 1
  - Par exemple : o(S) = 6 et o(S_g) = 0
  - On note δ(S) la distance entre la première et la dernière position instanciée ou fixée dans la chaîne ou encore longueur utile d 'un schema.
  - Par exemple : δ(00*010*1) = 8-1 = 7
  - Schema = outil de classification des similarités structurelles des chaînes.

Algorithmes Génétiques

- Théorème des Schemata : effet de la reproduction
  - Supposons qu 'il y ait m représentants d 'un schema S au temps t
  - m = m(S,t)
  - Reproduction : une chaîne est copiée avec $p_i = F_i / \Sigma F_i$
  - Taille population : n individus (constante)
  - En moyenne $m(S,t+1) = m(S,t).n.F(S)/\Sigma F_i$, avec F(S) la moyenne des valeurs d 'adéquation des chaînes représentants S à la date t.
  - Si on considère $F_{moy} = \Sigma F_i/n$ la moyenne des valeurs d 'adéquation dans la population, il vient : $m(S,t+1) = m(S,t).F(S)/F_{moy}$
  - Interprétation : un schema se dev à une vitesse égale au rapport de l 'adéquation moyenne du schema sur l 'adéquation moyenne de la population. Si $F(S)>F_{moy}$, le schema sera plus représenté à la génération suivante

Algorithmes Génétiques

- Théorème des Schemata : effet de la reproduction
  - Hypothèse : F(S) reste supérieur à la valeur moyenne de cF_{moy} avec c constante.
  - $m(S,t+1) = m(S,t).(F_{moy}+cF_{moy})/F_{moy} = (1+c).m(S,t)$
  - $m(S,t) = m(S,0).(1+c)^t$
  - Les schemata de valeur d 'adéquation supérieure à la moyenne augmentent de manière exponentielle (/ disparaissent de manière exponentielle).
  - Si on considère un individu de longueur L, il est le représentant de 2^L schemata au maximum dans la population (une valeur définie ou indéfinie par position). Une population de n individus représente donc au maximum $n.2^L$ schemata.
  - Reproduction = traitement en // de $n.2^L$ schemata maximum (//isme intrinsèque de Goldberg)

Algorithmes Génétiques

- Théorème des Schemata : effet du croisement

4-5

$C = \quad 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1$
$S_1 = \quad *\ 1\ *\ *\ *\ *\ 0\ *$
$S_2 = \quad *\ *\ *\ *\ 0\ 0\ *\ *$

  - C est un représentant de S_1 et de S_2
  - Hypo : C participe à un croisement avec C '
  - Si le point de coupure intervient entre 4 et 5 (0111 | 0001) S_2 sera toujours représenté
  - Par contre ce n 'est pas obligatoire dans le cas de S_1
  - 7 positions de coupure possibles
  - S_2 peut être détruit avec une probabilité proche de 1/7 (entre les 0)
  - S_1 peut être détruit avec une probabilité proche de 5/7
  - $\delta(S_2) = 1$ et $\delta(S_1) = 5$

Algorithmes Génétiques

- Théorème des Schemata : effet du croisement

  - évaluation de la probabilité de survie / destruction d 'un schema
  - $P_{destruction}(S) = \delta(S) / (L-1)$
  - $P_{survie}(S) = 1- \delta(S) / (L-1)$
  - Probabilité de croisement $P_{croisement}$
  - $P_{survie}(S)>1-P_{croisement}(S). \delta(S) / (L-1)$
  - Hypothèse d 'indépendance des probabilités de reproduction & croisement :

$$m(S,t+1) \geq m(S,t).\frac{F(S)}{Fmoy}.(1-(Pcroisement.\frac{\delta(S)}{L-1}))$$

  - Plus l 'adéquation moyenne d 'un schema est élevée, plus sa distance définie est faible et plus le schema sera représenté dans la population suivante

Algorithmes Génétiques

- Théorème des Schemata : effet de la mutation
  - Soit $P_{mutation}$ la probabilité de mutation d 'un bit
  - La probabilité de survie d 'un schema est proche de $(1-P_{mutation})^{o(S)}$
  - $P_{mutation}$ faible, approximation : $P_{survie}(S) \sim 1-P_{mutation}.o(S)$

$$m(S,t+1) \geq m(S,t).\frac{F(S)}{Fmoy}.(1-(Pcroisement.\frac{\delta(S)}{L-1})-(o(S).Pmutation))$$

  - Théorème des Schemata / briques élémentaires
  - L 'algorithme génétique recherche des structures ' petites ' et très pertinentes pour adapter la solution de manière croissante exponentielle.
  - Stratégie optimale en terme d 'essais/erreurs (*K-Armed Bandit*)

Algorithmes Génétiques

- Remarques
  - Algorithmes Génétiques + compliqués (ici, exemple simple)
  - Taille variable population
  - Taille variable des chaînes
  - Modification du codage des chaînes
  - Gènes ' sauteurs '
  - Reproduction sexuée
  - Mise à l 'échelle fonction d 'adéquation
  - Variations de la fonction d 'adéquation
  - Haploïdie / Diploïdie
  - Dominance / récession des gènes
  - // ismes et niches
  - Modification de stratégies
  - Terminaison
  - etc...

Algorithmes Génétiques

- 4. Quelques Applications
  - Genetic Programming
  - Reconnaissance d ’écriture
    - (Ménier/Lorette)
  - Modèles d ’adaptation en animation comportementale

Algorithmes Génétiques

- Programmation génétique : Koza
  - Principe : le Génotype est un programme
  - Phénotype = sémantique du programme
  - On fait évoluer des populations de programmes pour fabriquer le programme qui correspond au comportement souhaité
  - Adéquation = valeur d ’évaluation d ’un programme
  - La machine doit trouver un algorithme
  - Exemple des fourmis...

Algorithmes Génétiques

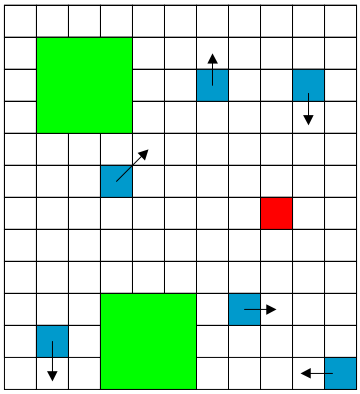
- Programmation génétique

Une fourmi (déplacement initial aléatoire)

Le nid des fourmis (position aléatoire)

Nourriture (position aléatoire)

tore

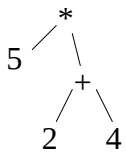


Colonie de fourmis artificielles qui doivent transporter de la nourriture vers leur nid. Chaque fourmi dépose sur son passage des phéromones pour indiquer le chemin à d 'autres individus (atténuation / temps). Problème : trouver l 'algorithme optimal qui régit le comportement d'une fourmi pour que la colonie survive...

L'ordinateur ne doit pas optimiser une solution, il doit trouver seul l 'algorithme que doit suivre chaque fourmi.

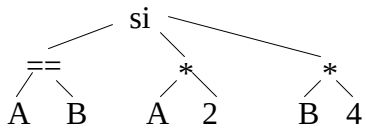
Algorithmes Génétiques

- Programmation génétique



  - codage algorithme : arbre (à-la-lisp)
  - exemple : 5*(2+4)
  - exemple : si (A==B) alors rendre (A*2) sinon rendre (B*4)

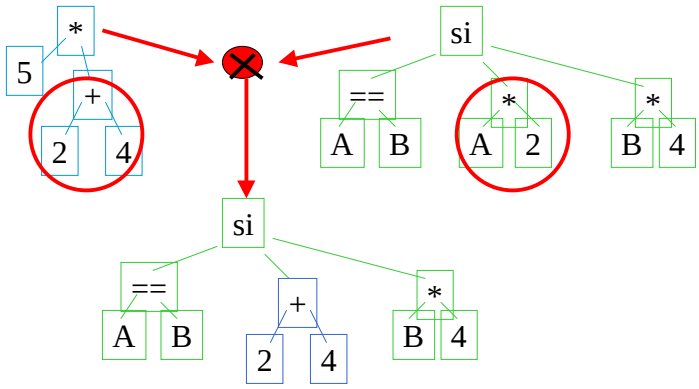
```
(si
        (== A B)
        (* A 2)
        (* B 4)
)
```



  - (Progn eval1 eval2 … evaln)
  - Chaîne génétique = un arbre = un programme

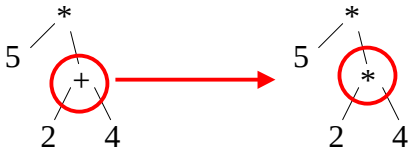
Algorithmes Génétiques

- Programmation génétique
  - Croisement entre deux solutions : échange de sous-arbres



Algorithmes Génétiques

- Programmation génétique
  - Mutation :
  - un nœud est tiré au hasard dans l 'arbre
  - Si c 'est un opérateur n-aire, on le remplace par un autre opérateur n-aire
  - Si c 'est une feuille, on la remplace par un autre element unaire
  - etc...



Algorithmes Génétiques

- Programmation génétique

  – Adéquation : nombre de nourritures transportées par la fourmi en un temps borné
  – Un cycle = simulation de la population pour 100 déplacements, puis reproduction (croisement, mutation)

  – Langage de commande dédié :
  – AVANCE_ALEATOIRE : modifie aléatoirement la direction de la fourmi et la fait avancer de deux cases
  – AVANCE_VERS_NID : avance la fourmi d 'une case vers son nid
  – PREND_NOURRITURE : la fourmi attrape la nourriture si elle est sur sa case
  – DEPOSE_PHEROMONE : la fourmi dépose une phéromone à la case où elle de trouve (nuage de 3x3 s 'atténuant dans le temps)

Algorithmes Génétiques

- Programmation génétique

  – SI_NOURRITURE : 2 arg, le premier est executé s 'il y a de la nourriture sur la case de la fourmi, le second sinon
  – SI_PORTE_NOURRITURE : idem, mais test différent
  – AVANCE_VERS_NOURRITURE_SINON : la fourmi vérifie s 'il y a de la nourriture dans une case adjacente et se déplace vers elle, sinon exécute l 'argument
  – AVANCE_VERS_PHEROMONE-SINON : idem mais avec PHEROMONE
  – PROGN : execution d 'arguments en cascade

  – Population de programmes initiale tirée au hasard

Algorithmes Génétiques

- Programmation génétique
  - 1ère génération : 93% des fourmis ne ramènent pas une seule nourriture au nid
  - 10ième génération : la meilleure fourmi rapporte 54 unités de nourritures au nid
  - 72 à la 20ième génération
  - 110 à la 30ième génération
  - A la génération 38, le programme résultant est présent sur chaque fourmi. La colonie ramène la totalité de la nourriture présente sur la carte (144 par fourmi)

Algorithmes Génétiques

- Programmation génétique

```
(PROGN   (PREND-NOURRITURE)
         (SI-PORTE-NOURRITURE)
                  (PROGN   (AVANCE-VERS-PHEROMONE-SINON
                                    (AVANCE-VERS-NOURRITURE-SINON
                                             (PREND-NOURRITURE)))
                           (AVANCE-VERS-NOURRITURE-SINON
                                    (PREND-NOURRITURE))
                           (AVANCE-VERS-NID)
                           (DEPOSE-PHEROMONE)
                           (AVANCE-VERS-NID)
                           (DEPOSE-PHEROMONE)
                           (AVANCE-VERS-NOURRITURE-SINON
                                    (SI-NOURRITURE
                                             (PREND-NOURRITURE)
                                             (AVANCE-VERS-PH-SINON)
                                                      (AVANCE-ALEA)))))))
```

Algorithmes Génétiques

- Programmation génétique - autres exemples
- Recherche symbolique d 'expression
  - Étant donné une courbe quelconque, trouver la fonction mathématique y=f(x)
  - Idem: codage de l 'expression sous forme d 'arbre
  - Opérateurs sin, cos, int, tan, *, - etc…
  - Distance quadratique entre la courbe générée et la courbe à approximatif
  - Calculer la fonction d 'adéquation pour minimiser la distance quadratique (ou maximiser son inverse)
- Adaptation du programme de déplacement d 'un robot (mars) en fonction d 'un changement de conditions (NASA)

Algorithmes Génétiques

- 1 Creation, using genetic programming, of a better-than-classical quantum algorithm for the Deutsch-Jozsa "early promise" problem B, F (Spector, Barnum, and Bernstein 1998)
- 2 Creation, using genetic programming, of a better-than-classical quantum algorithm for the Grover's database search problem B, F (Spector, Barnum, and Bernstein 1999)
- 3 Creation, using genetic programming, of a quantum algorithm for the depth-2 AND/OR query problem that is better than any previously published result B, D (Spector, Barnum, Bernstein, and Swamy 1999)
- 4 Creation of soccer-playing program that ranked in the middle of the field of 34 human-written programs in the Robo Cup 1998 competition H (Andre and Teller 1999)
- 5 Creation of four different algorithms for the transmembrane segment identification problem for proteins B, E (Koza, Bennett, Andre, and Keane 1999)
- 6 Creation of a sorting network (O'Connor, and Nelson 1962) for seven items using only 16 steps A, D (Koza, Bennett, Andre, and Keane 1999)
- 7 Rediscovery of the ladder topology for lowpass and highpass filters A, F (Koza, Bennett, Andre, and Keane 1999) (Campbell 1917)
- 8 Rediscovery of "*M*-derived half section" and "constant *K*" filter sections A, F (Koza, Bennett, Andre, and Keane 1999) (Zobel 1925, claim ---)
- 9 Rediscovery of the Cauer (elliptic) topology for filters A, F (Koza, Bennett, Andre, and Keane 1999) (Cauer 1934, 1935, 1936)
- 10 Automatic decomposition of the problem of synthesizing a crossover filter A, F (Koza, Bennett, Andre, and Keane 1999) (Zobel 1925, claim ---)

gildas.menier@univ-ubs.fr - Algorithmes Génétiques et apprentissage

Algorithmes Génétiques

- 11 Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits A, F (Koza, Bennett, Andre, and Keane 1999) (Darlington 1953)
- 12 Synthesis of 60 and 96 decibel amplifiers A, F (Koza, Bennett, Andre, and Keane 1999)
- 13 Synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions A, D, G (Koza, Bennett, Andre, and Keane 1999)
- 14 Synthesis of a real-time analog circuit for time-optimal control of a robot G (Koza, Bennett, Andre, and Keane 1999)
- 15 Synthesis of an electronic thermometer A, G (Koza, Bennett, Andre, and Keane 1999)
- 16 Synthesis of a voltage reference circuit A, G (Koza, Bennett, Andre, and Keane 1999)
- 17 Creation of a cellular automata rule for the majority classification problem that is better than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans D, E (Andre, Bennett, and Koza 1996)
- 18 Creation of motifs that detect the D–E–A-D box family of proteins and the manganese superoxide dismutase family C (Koza, Bennett, Andre, and Keane 1999)
- 19 Synthesis of analog circuit equivalent to Philbrick circuit (Philbrick 1956) A (Koza, Bennett, Keane, Yu, Mydlowec, and Stiffelman 1999)
- 20 Synthesis of NAND circuit A (Bennett, Koza, Keane, Yu, Mydlowec, and Stiffelman 1999)

Algorithmes Génétiques

- 21 Synthesis of digital-to-analog converter (DAC) circuit A (Bennett, Koza, Keane, Yu, Mydlowec, and Stiffelman 1999)
- 22 Synthesis of analog-to-digital (ADC) circuit A (Bennett, Koza, Keane, Yu, Mydlowec, and Stiffelman 1999)
- 23 Synthesis of topology, sizing, placement, and routing of analog electrical circuits G (Koza and Bennett 1999)
- 24 Synthesis of topology for a PID type of controller A, F (Koza, Keane, Yu, Bennett, Mydlowec 2000) (Callender and Stevenson. 1939)
- 24 Synthesis of topology for a controller with a second derivative A, F (Koza, Keane, Yu, Bennett, Mydlowec 2000) (Jones 1942)

- Quelques Sites intéressants :
- Site : www.genetic-programming.org
- chez Koza : http://smi.stanford.edu/~koza/

Bibliographie

- « Artificial Life:Spontaneous Emergence of Self-Replicating and Evolutionnary self-improving Computer Programs » J.R.Koza - Artificial Life III - Addison-Wesley 1994
- « Genetic Programming : A paradigm for Genetically Breeding Populations of Computer Programs to solve Problems » Goldberg Tech Report STAN-CS-90-1314 Stanford University, 1990
- « Genetic Algorithms in Search, Optimization and Machine Learning » Goldberg Addison-Wesley 1989
- « Genetic Algorithm and Walsh Functions » Goldberg Complex Systems 3, 1989
- « Learning with Genetic Algorithms: an Overview » Machine learning 3, P.121-138, 1988
- « Optimization by Simulated Annealing » S.Kirkpatrick, Science 220 P.671-680, 1993
- "Evolving Virtual Creatures," Sims, K., Computer Graphics (Siggraph '94)Annual Conference Proceedings, July 1994, pp.43-50.

Karl Sim (!)

Evolving Virtual Creatures 94 (Video)

gildas.menier@univ-ubs.fr - Algorithmes Génétiques et apprentissage