Méta-heuristiques ?

- Une des difficultés de la recherche heuristique est de trouver une heuristique appropriée
- L'heuristique dépend beaucoup du domaine d'application et son développement demande souvent des compétences avancées
- On peut donc s'intéresser à des méthodes heuristiques globales
  - proposant une méthode de recherche générique
  - dépendant aussi peu que possible du domaine d'application
- On parle alors de *méta-heuristique*
- Les *algorithmes génétiques* font partie de cette famille d'algorithmes.

Algorithmes génétiques

- Développés par John Holland (Université du Michigan) à partir des années 1960
- Les algorithmes génétiques s'inspirent
  - de la théorie de l'évolution de Darwin
  - de la génétique

  pour proposer une méthode de recherche et/ou d'optimisation.
- Ce qu'on cherche ici est une *solution* à un problème et non un *chemin* vers cette solution

  (Même si la solution peut être un chemin !)

Darwin et la sélection naturelle

- Charles Robert Darwin (Grande-Bretagne, 1809–1882)
- Publie en 1859 “The origin of species” qui pose les bases de la théorie de la sélection naturelle
- Ses idées n’allaient à l’époque pas de soi :

  > « La conclusion fondamentale à laquelle nous sommes arrivés dans cet ouvrage, à savoir que l’homme descend de quelque forme d’organisation inférieure, sera, je le regrette de le penser, fort désagréable à beaucoup de personnes. » (Charles Darwin, la “Descendance de l’homme et la sélection naturelle”)

Sélection naturelle – idées principales

- Les individus possèdent un certains nombre de caractéristiques héréditaires
- Ces caractéristiques peuvent évoluer (aléatoirement, ou en tout cas de manière *non orientée*)
- Un environnement à ressources limitées provoque une sélection, qui favorise la survie des individus *les plus adaptés*
  - Sélection de survie
  - Sélection sexuelle
- Darwin n'a pas expliqué le mode de transmission de caractères héréditaires
  - Notamment, il n'a pas exclu l'hérédité des caractères acquis !

Mendel et la génétique

- Johann Gregor Mendel (Autriche, 1822–1884)
- Considéré comme le père de la génétique
- A vécu à la même époque que Darwin, mais il semble qu'il n'aient jamais correspondu !
- Sur la base d'expériences botaniques, publie ses fameuses "lois de Mendel" qui décrivent les règles de l'hérédité
- Il postule notamment que l'hérédité est gouvernée par des "doubles commandes" chez les deux parents et qu'une seule commande est transmise à l'enfant
- Au cours du XXe siècle, on découvrira progressivement les chromosomes et l'ADN, qui permettent d'expliciter le support de l'hérédité

Un peu de terminologie

- Le support physique du matériel héréditaire est l'*acide désoxyribonucléique* (ADN), qui peut former des chaînes ("double hélice") de *nucléotides* (A,T,G,C)
- Une suite de nucléotides codant pour une protéine est appelée *gène*
- L'ensemble des gènes constitue le *génome*, dont le support physique est le *chromosome*
  - Certaines partie du génome sont *non-codantes* !

Le principe de base

- On choisit un code pour représenter les solutions d'un problème sous la forme de "chromosomes" (informatiques !)
- On génère (plus ou moins) au hasard une *population* représentant une partie du matériel génétique possible
- On fait ensuite évoluer cette population selon trois mécanismes :
  - La *sélection* qui a tendance à ne garder que les individus les plus adaptés (les meilleures solutions du problème)
  - Le *croisement*, qui combine les caractéristiques de deux chromosomes pour en donner un nouveau
  - La *mutation*, qui va modifier (plus ou moins) aléatoirement un ou plusieurs gènes d'un génome

Le codage

- Le codage est un élément important pour l'utilisation des algorithmes génétiques
- Il doit bien entendu permettre d'atteindre toutes les solutions !
- Il doit également se prêter aux opérations de croisement et de mutation
  - À noter que ces opérations n'ont pas besoin de correspondre à des changements "significatifs" du point de vue du problème...
  - ... tant que le gène obtenu code toujours une solution correcte !
- On en distingue habituellement trois types :
  - Le codage binaire
  - Le codage sur un alphabet
  - Le codage en arbre

Le codage binaire

- Codage utilisé à l'origine par John Holland
- Avantages
  - Maximise les possibilités de croisement et de mutation
  - Un bit donné ne représentant souvent rien de très significatif à nos yeux, l'algorithme est susceptible de trouver des solutions très originales !
  - Peut être traité de manière efficace et générique
- Désavantage majeur
  - Codage souvent peu naturel, parfois même difficile à mettre en place...

Codage sur un alphabet

- Il s'agit d'un code tout à fait standard : on se donne un alphabet de base pour exprimer les solutions du problème
- Il faut parfois définir un *langage* pour délimiter les chaînes significatives
- Les mutations et croisements doivent alors rester dans le langage !
- Souvent plus naturel que le codage binaire

Codage en arbre

- Plutôt que de représenter une solution sous forme de chaîne “à plat”, on la représente sous forme d’arbre
- Permet dans certains cas de simplifier le problème de rester dans le langage lors des croisements et mutations
- Peut permettre de mieux séparer des sous-problèmes dans des cas où la représentation est grande
  - “Si la première branche contient X, le contenu de la deuxième branche n’a pas d’importance. . . ”
  - À l’extrême, pourrait même servir à représenter des solutions infinies. . .

La sélection

- Pour faire évoluer la population, il faut “faire de la place”
- Pour ce faire, on va à chaque pas de l’algorithme sélectionner une sous-partie de la population qui servira de base à la population suivante
- Il existe de nombreuses manière de le faire, dont nous allons survoler les plus courantes

Les types de sélection

Sélection uniforme On choisit au hasard n individus. Pas très efficace

Sélection par rang On choisit toujours les n meilleurs individus. Efficace, mais risque de provoquer une convergence trop rapide vers un optimum local

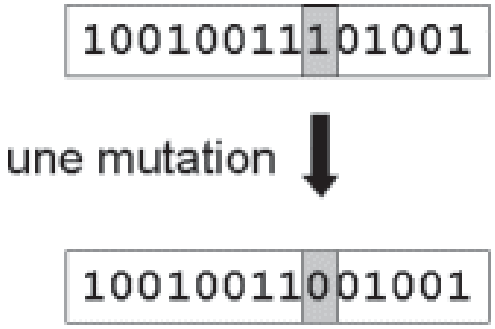
Sélection probabiliste Chaque individu a une probabilité d'être choisi proportionnelle à son degré d'adaptation

Sélection par tournois On choisit (uniformément ou non) des paires d'individus et on les fait "s'affronter" : le plus adapté sera choisi

- Chacun de ces types de sélection peut être enrichi d'un mécanisme *élitiste* : on garde toujours le meilleur individu (pour ne pas "régresser" dans notre recherche. . .)

Croisement et mutations génériques (codage binaire)

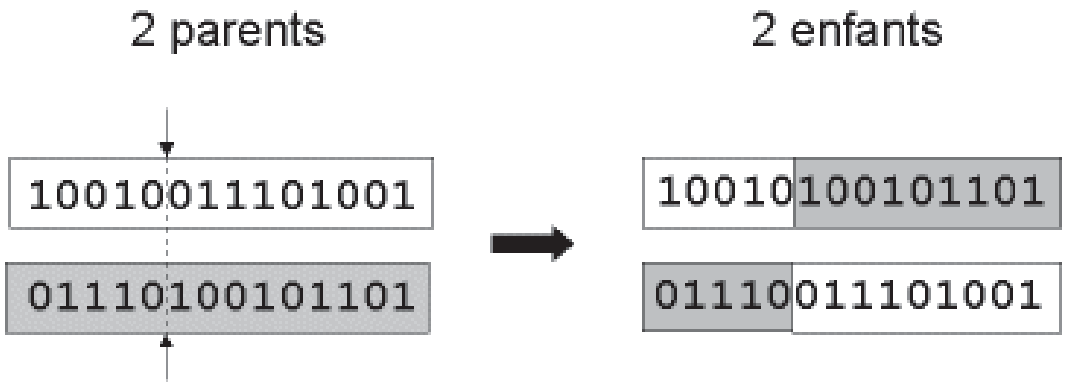
- Avec un codage binaire (ou tout codage d'alphabet A dont le langage est tout A^*), on peut définir des opérateurs de croisement et de mutation génériques
- Mutation :



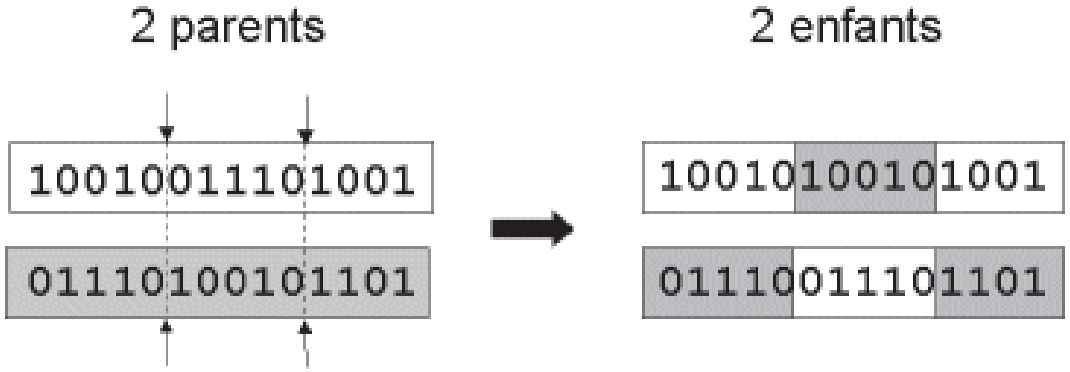
- Le croisement peut s'effectuer en 1 ou n points

Croisements en *n* points

- Croisement en un point :



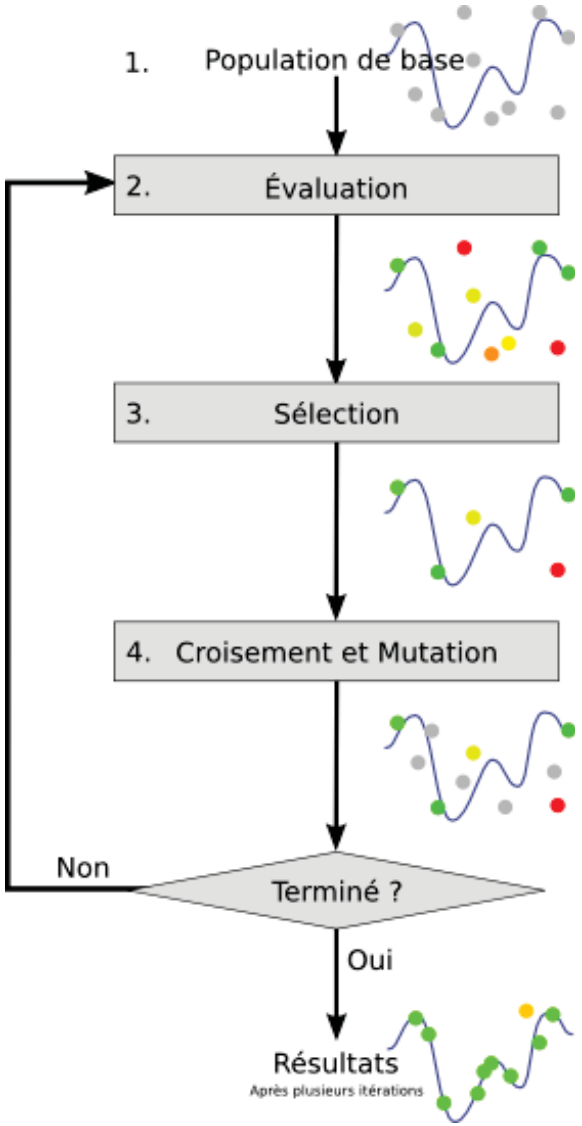
- Croisement en deux points



Croisements spécifiques

- Suivant le codage choisi, on peut être amené à développer des croisements et des mutations “ad hoc”
  - En particulier, le résultat du croisement doit toujours représenter une solution du problème !
- Exemple : le problème du voyageur de commerce
  - Un codage naturel est de représenter une solution par une suite de nombre représentant les villes dans l’ordre parcouru
  - Dans ce cas, les opérateurs de mutation et de croisement génériques *ne sont pas applicables* !
- Suivant les cas, on peut “optimiser” les opérateurs de croisement et/ou de mutation en ne les appliquant que s’ils amènent une amélioration (convergence plus rapide, mais exploration moins large. . .)

Vue d'ensemble



L'arrêt

- Une question délicate est : quand considère-t-on qu'on a terminé ?
- Un algorithme génétique est un algorithme "Anytime" : il peut fournir une réponse en tout temps, mais la *qualité* de la réponse augmente avec le temps
- Critères d'arrêt possible :
  - On a trouvé une solution *assez bonne*
  - Budget (temps ou calcul) épuisé
  - Nombre de générations maximal atteint
  - Pas d'amélioration dans la qualité des meilleures solutions sur n générations
  - Observation humaine
  - ...

Quelques considérations sur l'efficacité

- L'idée de base des algorithmes génétiques est de se reposer sur une évolution parfaitement aléatoire pour explorer librement l'espace des solutions
- La convergence vers une bonne solution est alors amenée par le processus de sélection
- Cette idée, proche des théories de l'évolution, permet de trouver des solutions originales, mais peut être très lente...
- On peut encourager une recherche plus rapide avec des opérateurs de mutation et de croisement un peu plus "orientés"
  - Il faut cependant se méfier des optima locaux...

Quand utiliser les algorithmes génétiques

- Les algorithmes génétiques peuvent trouver une bonne solution de manière efficace, mais il ne constituent bien entendu pas une solution universelle !
- Il sont adaptés quand
  - L'espace des solutions est grand (sinon, autant faire une recherche exhaustive...)
  - Il n'existe pas d'algorithme déterministe suffisamment efficace
  - On préfère une *assez bonne* solution *assez rapidement* plutôt que la solution optimale en un temps indéterminé

Quelques exemples d'application

- Le problème du voyageur de commerce et autres problèmes de recherche
  - Conception d'horaires, ...
- Motorola semble avoir utilisé les algorithmes génétiques pour développer automatiquement des suites de tests logiciels
  - Individu : jeu d'entrée
  - Valeur d'adaptation : portions de code testées
- Pilotage d'un robot (NASA pour Pathfinder, Sony pour Aibo, Aldebaran pour Nao, ...)

Les limites

- La principale difficulté des algorithmes génétiques est qu'ils sont très sensibles au choix des paramètres
  - Taille de la population
  - Taux de renouvellement
  - Taux de mutation
  - ...
- On est jamais sûr d'avoir trouvé la solution optimale !
- En particulier, on peut se retrouver bloqué dans un optimum local
  - Pour éviter cela, on peut essayer de toujours garder une certaine variété de population...

Proches parents

- La *programmation génétique* est basée sur les mêmes principes, mais on fait évoluer le code, pas les paramètres !
- Le *recuit simulé* utilise un seul individu qui parcourt l'espace des solutions, d'abord très librement puis de manière de plus en plus contrainte
- L'optimisation par colonie de fourmis (!) fait se promener des "fourmis" dans l'espace des solutions et utilise la diffusion de "phéromones" pour attirer d'autres fourmis dans les zones intéressantes
- …

Pour en savoir plus. . .

- http://fr.wikipedia.org/wiki/Algorithme_génétique
- http://magnin.plil.net/spip.php?rubrique8
- . . . Sans oublier une applet d'une redoutable efficacité pour résoudre le problème du voyageur de commerce : http://www.mac.cie.uva.es/~arratia/cursos/UVA/GeneticTSP/JAVASimultn/TSP.html