CHPS0742
Optimisation

Cours 5

Problèmes NP-Difficiles
Résolution par méthodes approchées

Pierre Delisle
Université de Reims Champagne-Ardenne
Département de Mathématiques et Informatique
Octobre 2018

Plan de la séance

- Algorithmes de résolution approchés
  - Méthodes basées sur un voisinage
    - Méthodes de descente
    - Recuit simulé
    - Méthode tabou
  - Méthodes basées sur une population
    - Algorithmes génétiques
    - Optimisation par Colonie de Fourmis

La fable des randonneurs ?!?!

Il était une fois...

- 4 randonneurs perdus en montagne, à 4 endroits différents, dans le brouillard
- Une équipe de secours passe régulièrement sur le point de plus basse altitude
- Avec leur boussole et leur altimètre, il doivent trouver le chemin menant aux secours
  - Mais ils ne voient que leur environnement immédiat
  - Arrivés à un croisement, ils doivent s'engager sur un chemin pour savoir s'il monte ou descend

1er randonneur : l'optimiste

- À chaque carrefour, il note l'altitude et la direction empruntée
- Il essaie tous les chemins possibles en revenant en arrière
- Quand il aura fini de tout explorer, il pourra revenir à l'altitude la plus basse avec la certitude qu'il est au bon endroit
- Il est encore en train de se promener dans la montagne...

2e randonneur : l'opportuniste

- À chaque croisement, il suit le premier chemin descendant qu'il trouve
  - Descendre pour aller en bas, quoi de plus normal !
- Il est arrivé à un croisement où tous les chemins remontent, il s'est donc dit que c'était le point le plus bas et il a attendu
- Le lendemain matin, avec le brouillard dissipé, il a réalisé que c'était un trou...

3e randonneur : le gambler

- À un croisement
  - Chemin descendant → il le prend
  - Chemin ascendant → il fixe un nombre et lance les dés
    - Inférieur au nombre, il le prend, sinon il prend le suivant
  - Plus la pente ascendante est forte, plus le nombre est petit
  - Plus le nombre de chemins descendants consécutifs empruntés est grand, plus le nombre est petit
- Rendu à un trou, il aura une chance de s'en sortir

4e randonneur : le cerveau

- À un croisement, il prend le chemin qui descend avec la plus forte pente
- S'il n'y a pas de chemin descendant, il prend celui qui monte avec la plus faible pente
- Pour éviter de reprendre le chemin descendant qu'il vient de monter et de tourner en rond
  - Il s'interdit de faire marche arrière en mémorisant les 7 dernières directions qu'il a utilisées
  - S'il entrevoit un chemin particulièrement intéressant, il se permet de lever l'interdiction

Morale de l'histoire

- L'optimiste n'a toujours pas trouvé son chemin
- L'opportuniste a souffert d'hypothermie mais a trouvé son chemin le lendemain matin
- Le gambler et le cerveau ont trouvé leur chemin
  - Mais l'histoire ne dit pas lequel est arrivé en premier...

Morale de l'histoire, Part. 2

- Pour beaucoup de problèmes d'optimisation combinatoire
  - Déterminer une solution exacte peut prendre un temps de calcul phénoménal
  - Même avec des méthodes exactes « évoluées », il n'est pas raisonnable de le faire (NP-Difficiles)
  - Il faut se contenter d'une solution approchée en espérant qu'elle soit la meilleure possible dans un temps acceptable
    - Compromis souvent difficile à déterminer

Morale de l'histoire, Part. 2

- Différentes méthodes permettent d'atteindre une solution approchée
- Certaines sont spécifiques à un problème
  - Heuristiques
- D'autres sont plus génériques et adaptables à divers problèmes
  - Méta-heuristiques (ou heuristiques générales)
  - Peuvent servir de guide à des heuristiques subordonnées

Formulation du problème (pour la suite)

- Pour simplifier la suite, on considère seulement les problèmes de minimisation
  - Minimiser $f(X)$
    - $X \in S$
  - Où S est un ensemble fini, mais de grand cardinal
  - Un élément de S s'appelle une *solution réalisable* (ou solution par abus de langage)
  - Un élément de S qui donne à f sa valeur minimum s'appelle une *solution optimale*

Méthodes de descente

ou

Méthodes d'amélioration itérative

Principe

- À partir d'une solution de départ X_0
  - On engendre une suite finie de solutions X_i déterminées de proche en proche…
  - … X_{i+1} étant calculée à partir de X_i de sorte à ce que X_{i+1} soit meilleure que X_i
  - Autrement dit : $f(X_{i+1}) < f(X_i)$ pour tout i
- Reste à déterminer comment engendrer une nouvelle solution à partir d'une solution donnée
  - Voisinage d'une solution

Voisinage d'une solution

- On définit le voisinage d'une solution à l'aide d'une *transformation élémentaire* (ou *locale*)
  - Opération permettant de changer une solution *X* de *S* en une autre solution *X'* de *S*
- Ne modifie que « faiblement » la structure de la solution sur laquelle on l'applique
- Sous-ensemble de l'ensemble de toutes les transformations
- Ex : 1111111 devient 1111011

Voisinage d'une solution

- Le choix de la transformation dépend
  - Du problème à traiter
  - De la vitesse de l'évaluation de ses conséquences
  - De sa capacité à générer tout l'ensemble *S* ou un sous-ensemble pertinent
  - Exemple : sur une chaîne de *n* bits
    - Transformation : changer un bit par son complémentaire
    - À partir de n'importe quelle solution initiale …
    - … on peut arriver à n'importe quelle configuration en au plus *n* changements « bien choisis »

Voisinage d'une solution

- Définition
  - Étant donnée une transformation locale …
  - … le voisinage $V(X)$ d'une solution X …
  - Est l'ensemble des solutions que l'on peut obtenir en appliquant à X cette transformation locale
- Le voisinage dépend donc de la transformation considérée
  - Il en existe plusieurs...

Quelques voisinages habituels

- Complémentation (remplacement)
  - Pour les solutions codées en chaîne de bits
  - On remplace un bit quelconque de la chaîne par son complémentaire

1 0 0 1 1 **1** 0 0 1

Devient

1 0 0 1 1 **0** 0 0 1

Quelques voisinages habituels

- Échange
  - Pour les solutions codées en chaîne de caractères
  - On intervertit les caractères situés en 2 positions données de la chaîne

A **B** C D **E** F G

Devient

A **E** C D **B** F G

Quelques voisinages habituels

- Insertion-décalage
  - Solutions codées en chaîne de caractères
  - On choisit 2 positions *a* et *b*, on insère à la position a le caractère situé en position *b*, puis on décale tous les caractères anciennement situés entre *a* (inclus) et *b* (exclus)

A B C D E **F** G

Devient

A B **F** C D E G

Quelques voisinages habituels

- Inversion
  - Solutions codées en chaîne de caractères
  - On choisit 2 positions *a* et *b* avec *b* > *a*, puis on inverse l'ordre d'écriture des caractères

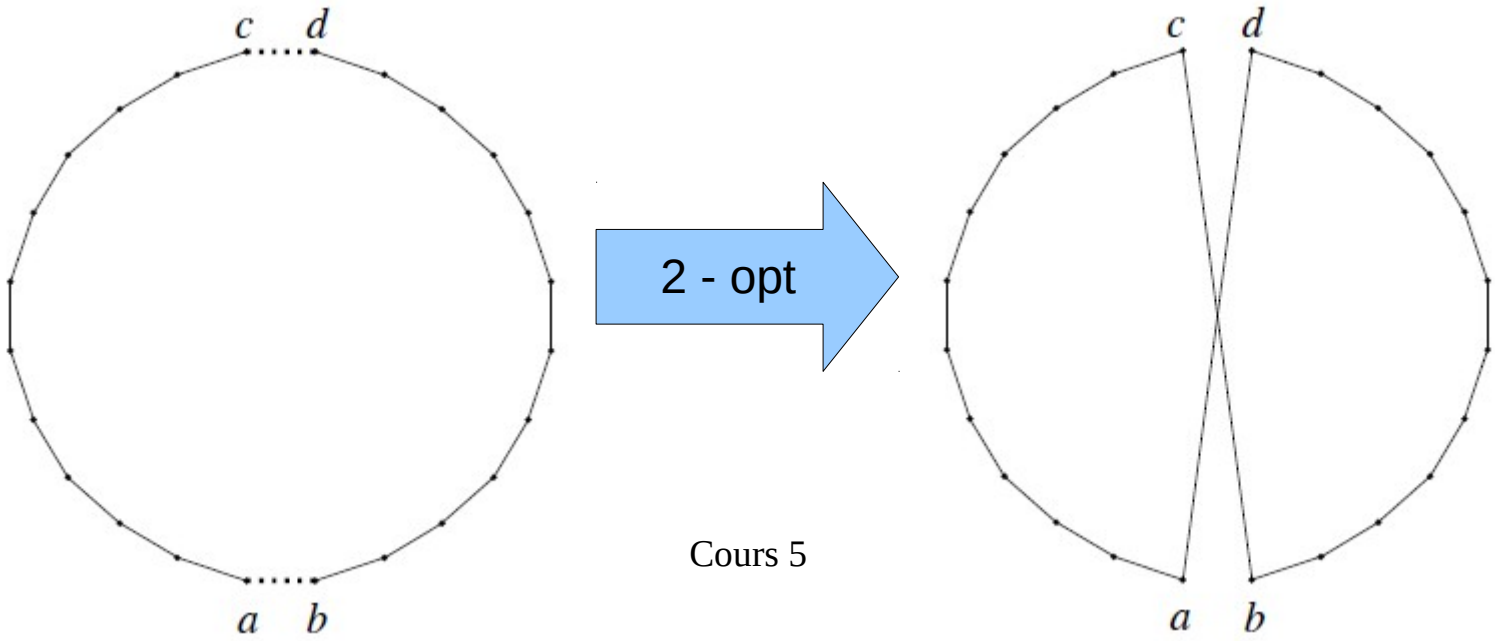
A **B C D E** F G

Devient

A **E D C B** F G

Pour le voyageur de commerce

- Transformation 2-Opt
  - On choisit 2 arêtes non adjacentes dans le cycle hamiltonien courant
  - On les remplace par les 2 arêtes qui permettent de reconstituer un cycle hamiltonien



Pour le voyageur de commerce

- Transformation 3-Opt
  - On choisit 3 arêtes non adjacentes dans le cycle hamiltonien courant
  - On les remplace par une combinaison de 3 arêtes qui permettent de reconstituer un cycle hamiltonien

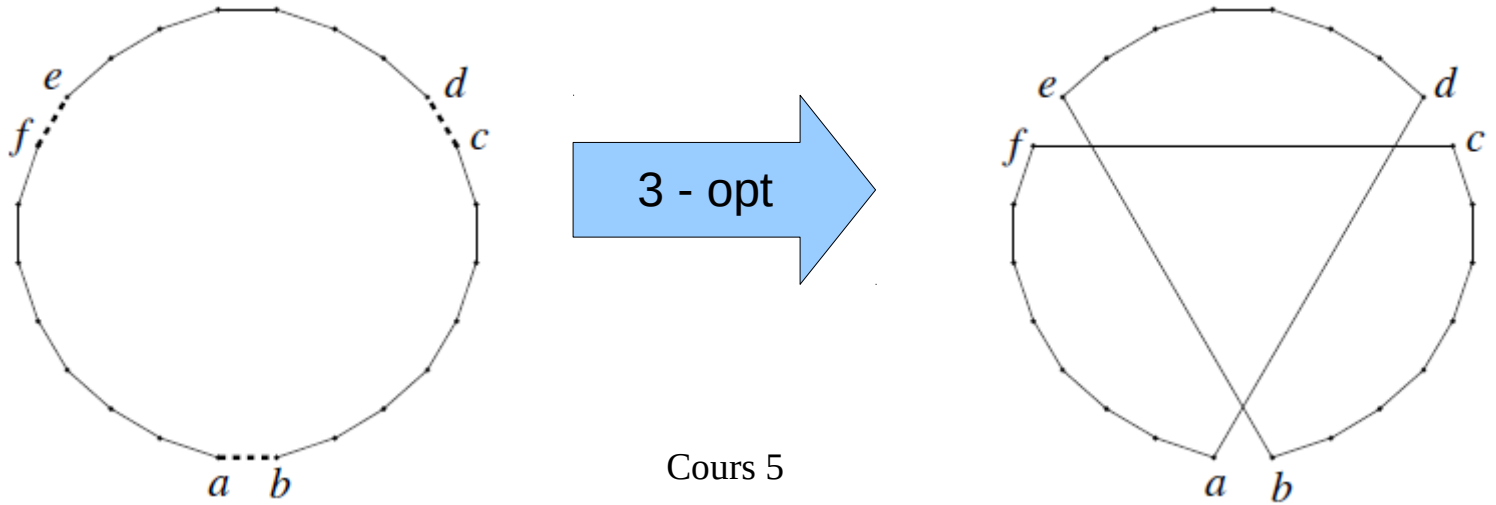


Schéma général d'une descente

- À partir d'une solution initiale X_0
  - On engendre une suite de solutions X_i vérifiant
  - $X_i \in V(X_{i-1})$ avec $f(X_i) < f(X_{i-1})$
- On arrête la méthode quand on atteint une configuration X_q telle que
  - $\forall\, X \in V(X_q),\ f(X) \geq f(X_q)$
- Autrement dit, on arrête quand aucun voisin ne peut apporter une diminution de f
  - X_q : *minimum local* de f par rapport à la transformation

Schéma général d'une descente

Engendrer la configuration de départ X
Répéter
    S'il existe $X' \in V(X)$ tel que $f(X') < f(X)$
        $X \leftarrow X'$
Jusqu'à ce qu'on ait $f(X') \geq f(X)$ pour tout $X' \in V(X)$
Retourner X comme solution finale

- Exploration du voisinage (temps de calcul variable)
  - Aléatoire
  - Premier meilleur voisin trouvé
  - Meilleur voisin de k voisins trouvés
  - Meilleur voisin de tout le voisinage

Schéma général d'une descente

- Génération de la solution de départ
  - Aléatoire
  - Algorithme glouton
  - ...
- Possibilité de faire des redémarrages sur des solutions initiales différentes (multi-start)
- De façon générale, une descente
  - Peut donner « rapidement » de bonnes solutions
  - Mais le minimum local peut être loin du minimum global (comme le deuxième randonneur...)

Algorithmes gloutons

Algorithmes gloutons

- En anglais → *greedy algorithms*
- À chaque itération
  - On fixe la valeur d'une (ou plusieurs) variables (on la « mange »)
  - Sans remettre en cause les choix précédents
- Méthode « proche en proche »
- Exemples
  - Arbre couvrant de poids minimum → Kruskal
  - Voyageur de commerce → plus proche voisin

Autres méthodes basées sur un voisinage

Autres méthodes basées sur un voisinage

- Recuit simulé
  - En anglais → *Simulated Annealing*
  - On simule un système physique chauffé à très haute température, puis graduellement refroidi pour atteindre une structure moléculaire stable
  - Descente « améliorée » → 3e randonneur
- Méthode tabou
  - On se déplace de solution en solution en interdisant de revenir à une configuration déjà rencontrée
  - Mémorisation des interdictions (liste « taboue »), possibilité de ne pas en tenir compte (critère « d'aspiration ») → 4e randonneur
- On y reviendra plus tard (si on a le temps)

Méthodes basées sur une population

Algorithmes génétiques

La théorie de l'évolution de Darwin

- Les êtres vivants évoluent sous l'effet du milieu
  - Ceux qui sont les mieux adaptés ont plus de chances de survivre et de se reproduire
- De génération en génération
  - Les caractéristiques des individus les mieux adaptés ont plus de chances de se répandre dans la population
- Hérédité
  - Transmission de ces caractéristiques

Hérédité

- Gène
  - Production des caractères héréditaires
  - Segment de *chromosome*, filament d'ADN, matériel génétique de toutes les cellules
- Toutes les cellules, sauf les cellules sexuelles
  - Possèdent le même nombre de chromosomes, présents sous formes de paires
  - Dans chaque paire, un des chromosomes vient du père et l'autre de la mère
- Cellules sexuelles
  - Ne fournissent qu'un seul chromosome

Fabrication de nouvelles cellules

- Division cellulaire (reproduction asexuée)
  - Une cellule duplique son matériel génétique avant de se couper en deux copies conformes
  - De temps en temps, une erreur de duplication se produit → mutation du gène
- Procréation (reproduction sexuée)
  - 2 parents interviennent pour fabriquer un enfant
  - L'individu n'est pas une reproduction de ses parents
  - Cellules sexuelles transmises par les parents apportent chacune la moitié des chromosomes de l'enfant

Crossing-over (enjambement)

- Lors de la production des chromosomes que les parents transmettent à l'enfant
  - Il arrive qu'un échange s'opère entre les chromosomes
  - Une séquence de l'un se substitue à une séquence de l'autre → crossing-over
- Accroît la diversité des individus

Algorithmes génétiques

- J.H. Holland, 1975
- S'inspirent (plus ou moins fidèlement) des mécanismes de l'évolution
- Considèrent
  - Une *population* de solutions X_1, X_2, X_3, … X_p de S
  - Appelés *individus*
  - (plutôt qu'une seule solution/configuration à la fois comme les méthodes précédentes)
  - $p \rightarrow$ taille de la population

Algorithmes génétiques

- Une solution X_i
  - Est représentée et manipulée sous forme de chaîne de caractères
  - Appelée *chromosome*
  - appartenant à un certain alphabet
- Les caractères formant un chromosome
  - Sont appelés *gènes*

Algorithmes génétiques

- 3 opérateurs
  - Sélection, Croisement, Mutation
- Vont permettre l'évolution de la population
  - Création de nouveaux individus construits à l'aide des anciens individus
  - Pendant un certain nombre de *générations*
- Pour appliquer la méthode
  - Il faut déterminer un *codage* adéquat pour une solution

Codage

- Problèmes de valeur
  - Meilleures valeurs possibles des gènes
  - Pour le problème du sac à dos, on peut représenter chaque variable x_i par un gène {0, 1}
  - 11001 → on prend les objets x_1, x_2 et x_5
- Problèmes de position
  - L'ensemble des valeurs est fixé à l'avance et on cherche la meilleure position pour chaque valeur
  - Voyageur de commerce → ville : gène {1, n}

Sélection

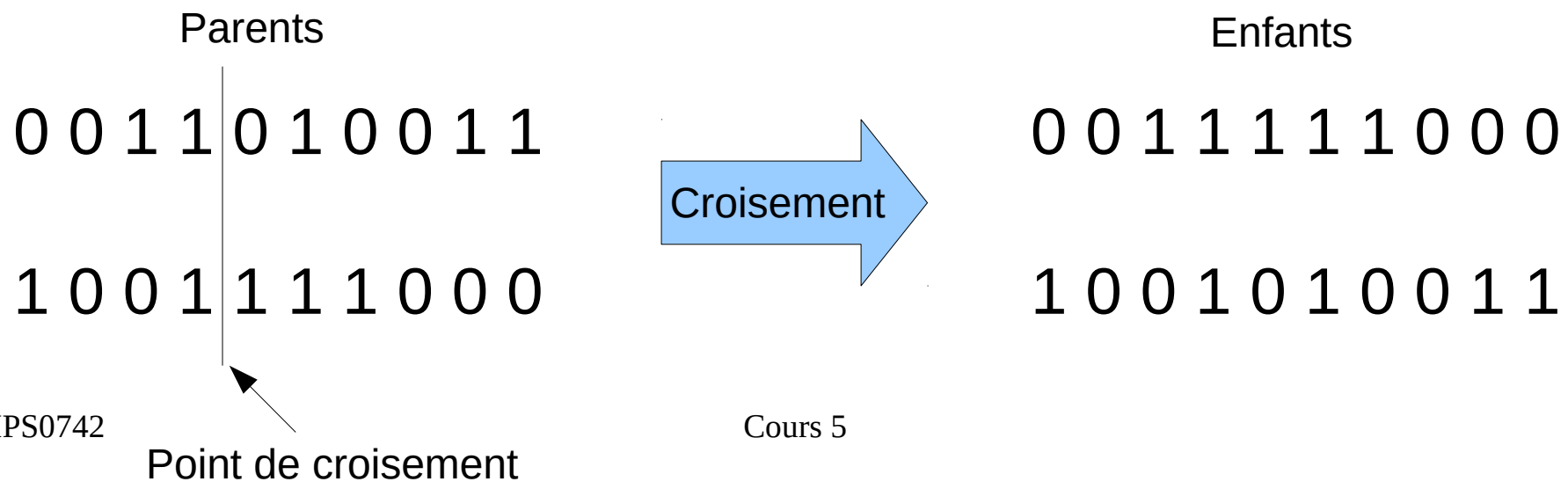
- Détermination des individus qui vont se reproduire dans la population
  - Dépend de la fonction *f* à optimiser (minimiser pour la suite)
  - La probabilité de choisir l'individu X_i sera plus grande si $f(X_i)$ est faible
- Exemple : roue de la fortune (roulette-wheel)
- Conserve une partie de la population pour la reproduction, les autres disparaissent

Croisement

- Recombinaison des chromosomes de deux parents procréateurs pour produire les enfants
  - Inspiré du mécanisme de « crossing-over »
  - On extrait une partie du code de chacun des parents et on réorganise ces parties
- Il existe plusieurs opérateurs de croisement
  - Croisement à un point (1-point crossover)
  - Croisement à deux points (2-point crossover)
  - ...

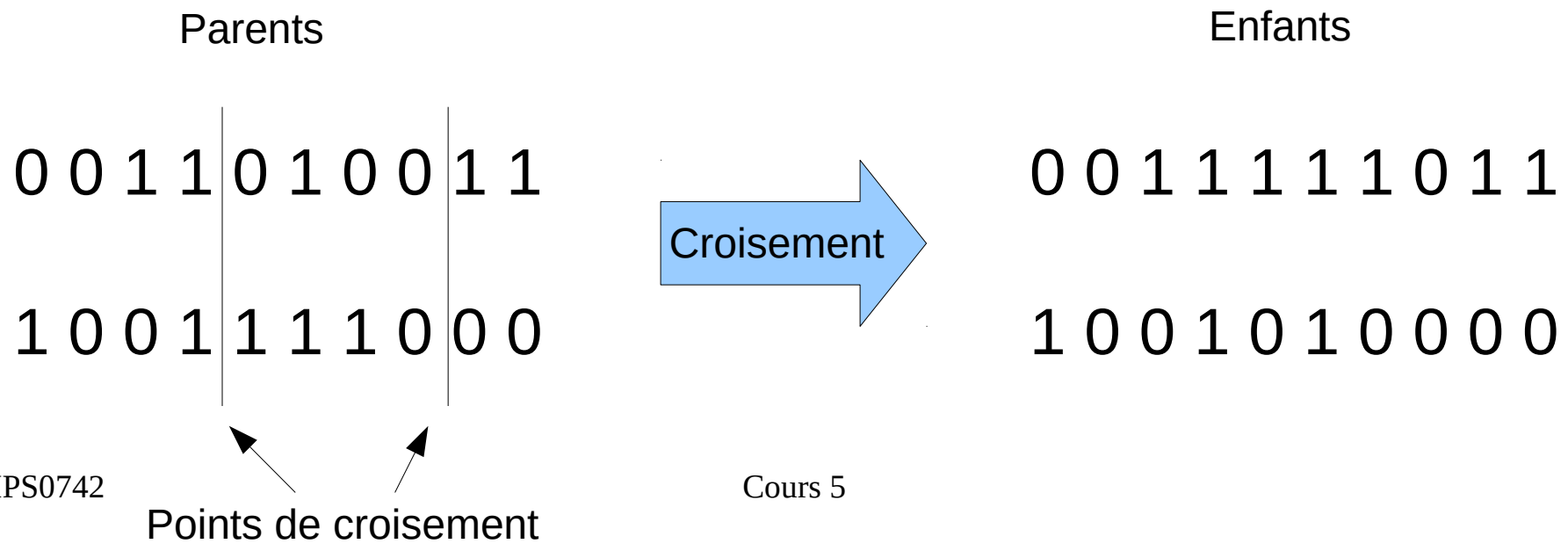
Croisement à un point (1-point crossover)

- On détermine aléatoirement une position (gène)
- On coupe chaque individu sur cette position
  - On obtient 4 morceaux de chromosomes
  - On forme 2 enfants en échangeant les morceaux



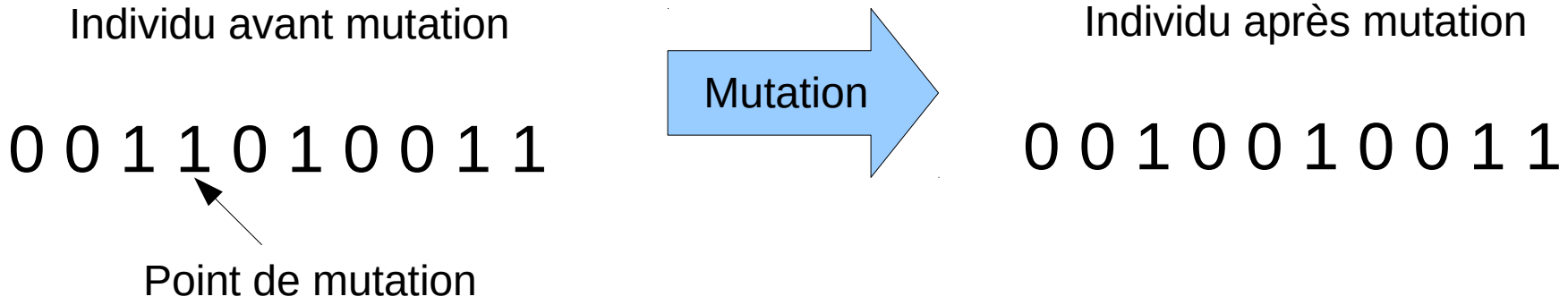
Croisement à deux points (2-point crossover)

- Deux positions plutôt qu'une
- On coupe chaque individu sur ces positions
- On échange les morceaux de chromosomes compris entre ces points de coupure



Mutation

- Modification sur le codage d'un invididu
  - Peut être aléatoire ou « plus intelligent »
  - But : apporter une diversité supplémentaire à la population et éviter une convergence prématurée
- Exemple : remplacer un gène par son complémentaire



Mutation

- Pour des problèmes de position
  - On ne peut changer la valeur d'un gène (doublon)
  - On peut appliquer une transformation locale
  - Exemple : 2-opt pour le voyageur de commerce
- On peut aussi développer des opérateurs de mutation spécifiques aux problèmes étudiés
- Pour éviter de trop faire « errer » la population, on fixera généralement une probabilité adéquate d'application de la mutation

Pseudo-code d'un algorithme génétique de base

Générer aléatoirement une population
TANT QUE Nb générations < Nb générations maximum

Évaluation :	Assigner une valeur d'adaptation (fitness Value) à chaque individu
Sélection :	Établir de façon probabiliste les paires d'individus qui vont se reproduire en accordant une meilleure chance aux meilleurs individus
Reproduction :	Appliquer les opérateurs de croisement aux paires sélectionnées
Mutation :	Appliquer un opérateur de mutation, avec une certaine probabilité, sur certains individus

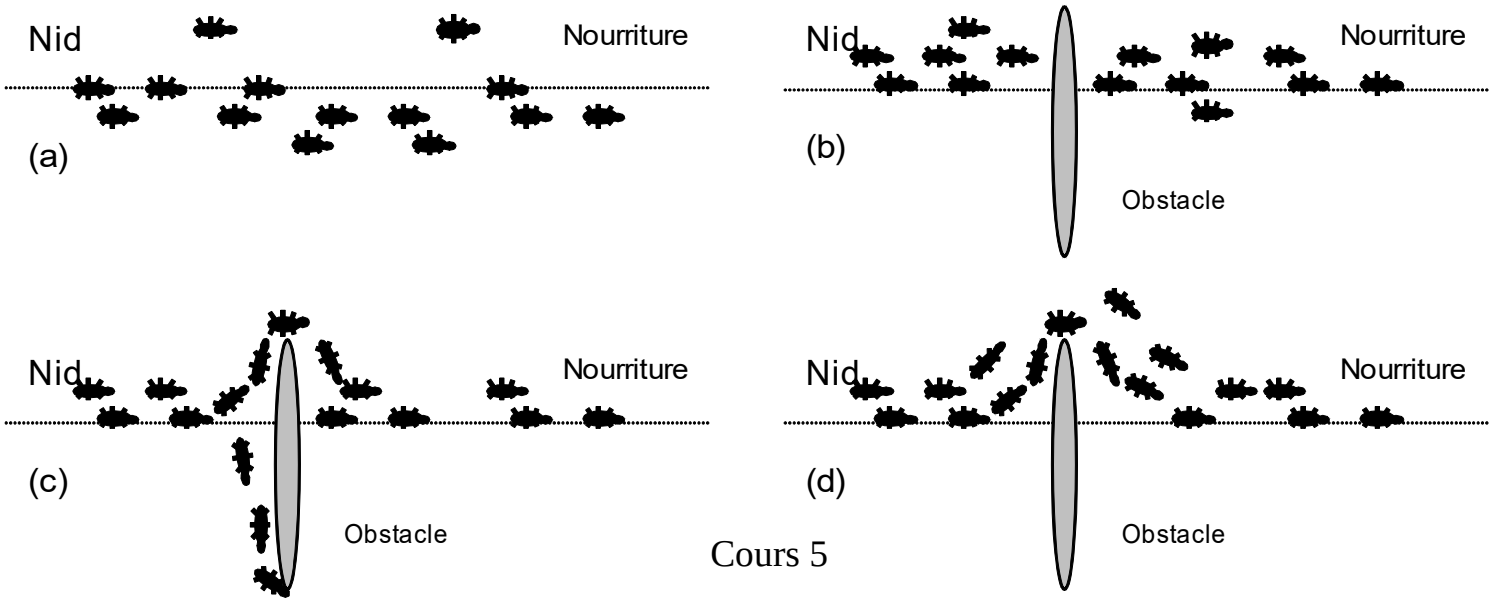
Synthèse

- La description donnée est un algorithme génétique « de base »
  - Il existe une multitude de variantes pour la sélection, le croisement et la mutation
  - Compromis entre le temps de calcul et la qualité des individus qu'il permet d'obtenir
- Exemple : Voyageur de commerce

Optimisation par Colonie de Fourmis

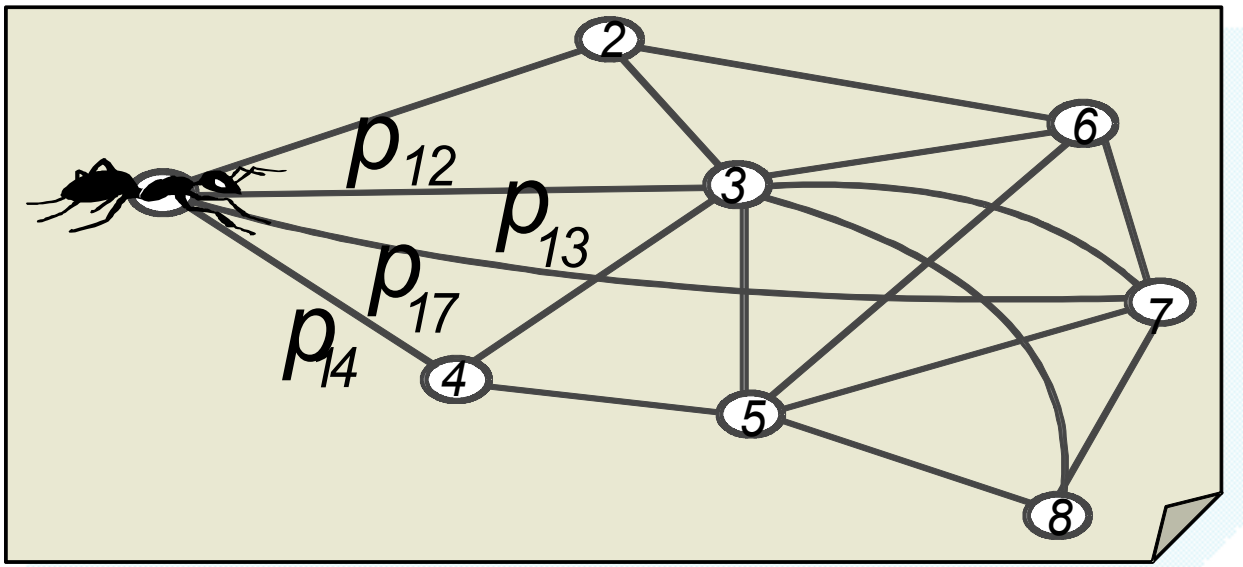
Optimisation par Colonie de Fourmis (OCF)

- Introduits par M. Dorigo en 1991
- Basé sur le comportement collectif des fourmis dans la recherche de nourriture
- Communication par la phéromone



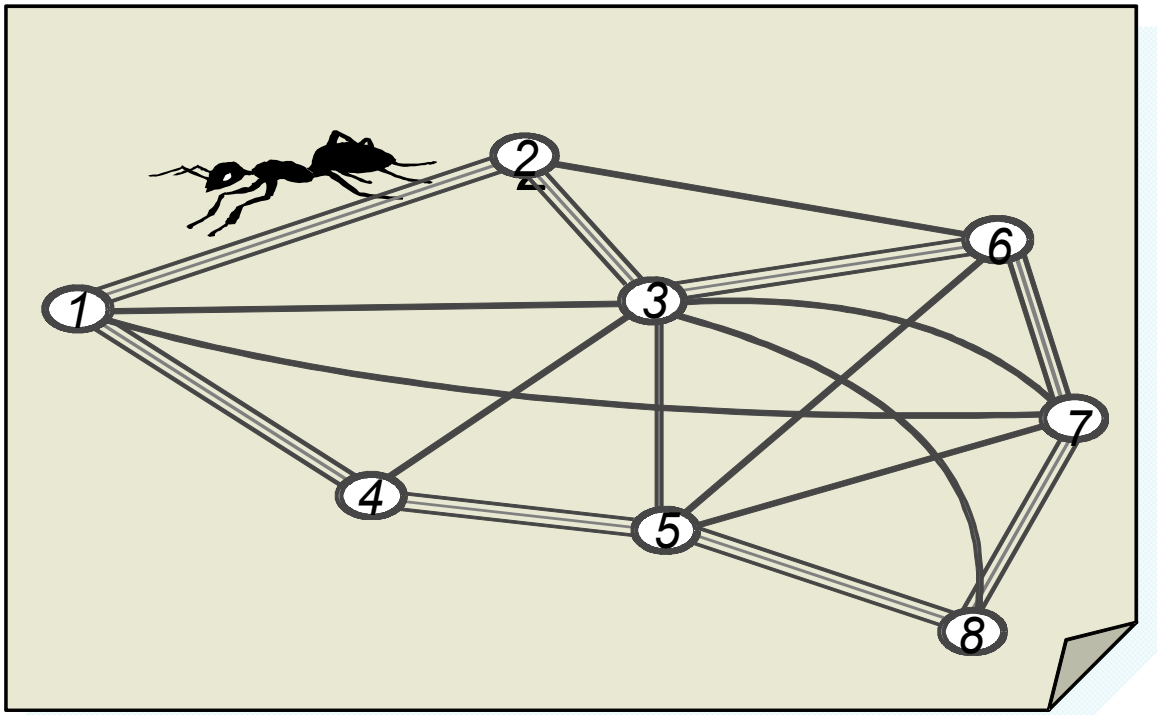
OCF pour VC : principe général

- La première fourmi sélectionne la prochaine ville à visiter
- Probabilité d'aller vers *j* : p_{ij}^{k}
- Ce choix n'est pas encore affecté par la trace



OCF pour VC : principe général

- Un tour est complété
- La trace est mise-à-jour
- La probabilité d'aller vers chacune des villes est alors influencée par la trace
- Plusieurs fourmis agissent en même temps
- Trace ↓avec le temps et ↑avec le passage des fourmis

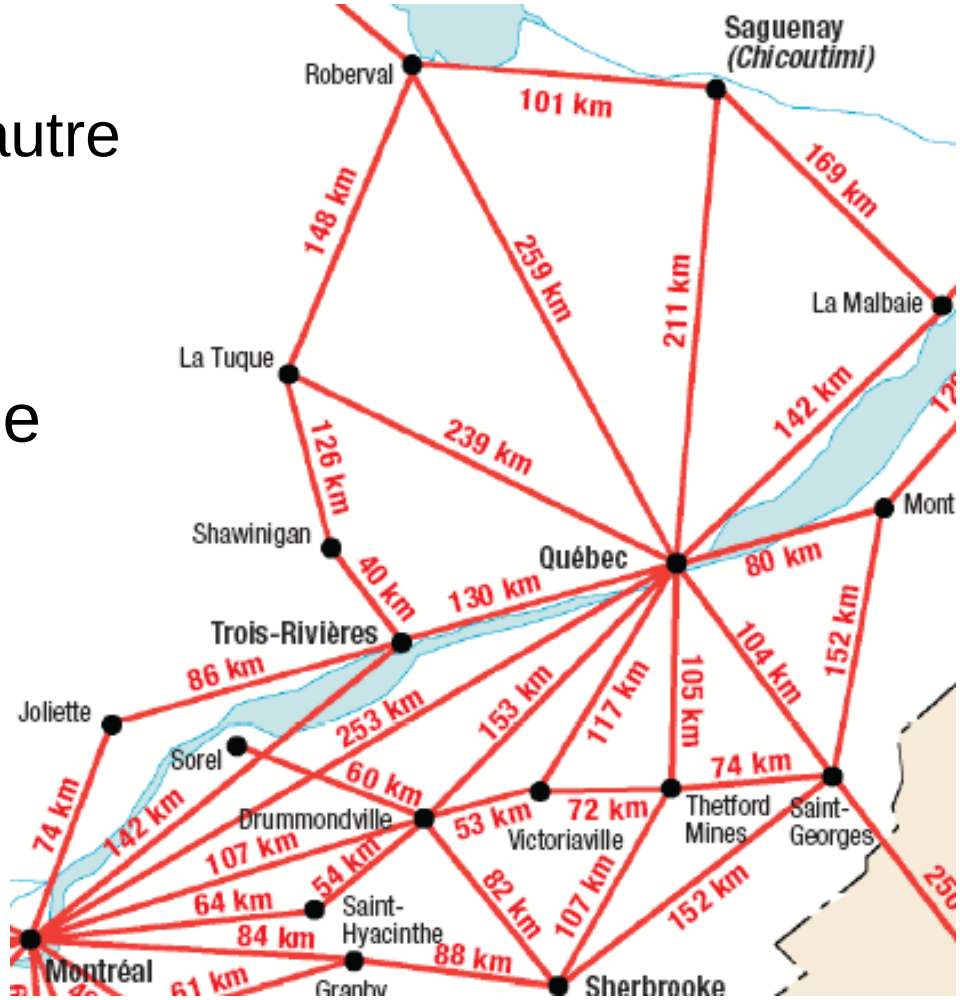


OCF pour VC : principe général

- Les fourmis
  - Se déplacent d'une ville à l'autre
  - Construisent des solutions
- La phéromone apporte une forme d'apprentissage globale

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}$$

- $\eta_{ij} = 1 / d_{ij}$ (distance)
- τ_{ij} = phéromone



Algorithme OCF

Initialiser l'information heuristique;
Initialiser les paramètres;
Initialiser la matrice de phéromone pour chaque paire de villes ij;
Placer les m fourmis sur une ville de départ différente;
POUR $t = 0$ à t_{Max} FAIRE
  *** Construction des solutions***
  POUR $i = 1$ à n FAIRE
    POUR $k = 1$ à m FAIRE
      Choisir la prochaine ville j à visiter selon la règle de transition;
  *** Évaluation de la tournée ***
  POUR $k = 1$ à m FAIRE
    Calculer la distance L_k de la tournée de la fourmi k;
  Mise à jour de $\Delta\tau_{ij}$
  POUR chaque arc (i, j)
    POUR $k = 1$ à m FAIRE
      Mettre à jour $\Delta\tau_{ij}^k$;
      $\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^k$;
  *** Mise à jour de la phéromone ***
  POUR chaque arc (i, j)
    Mettre à jour τ_{ij} à l'aide de $\Delta\tau_{ij}$;
    Réinitialiser $\Delta\tau_{ij}$;
  *** Mise à jour de la meilleure solution connue ***
Mettre à jour T^*;

La semaine prochaine

Applications et compléments