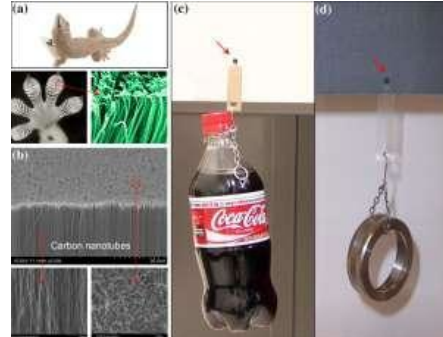


S8 - BE

Introduction aux systèmes collaboratifs  
(BE Colonie de Fourmis)  
Ecole Centrale de Lyon  
2016-2017

Alexander Saidi

# I.1 Intelligence Emergente



● De gauche à droite (source <http://sciencenordic.com>) :

- Les oiseaux et les avions
- Le scratch (la Velcro est une marque) a inspiré de la plante *bardane* (plante avec beaucoup de propriétés).
- Toiles d'araignée artificielle pour des gilets pare-balles
- Un super adhésif inspiré des *geckos* a créé des nanotubes de carbone pour collage (ici une bouteille de 650 gr sur une bande de 4x4mm , ...)
- La peau des requins contient un motif spécial qui permet une circulation rapide d'eau et empêche l'encrassement.
- Les bateaux / nageurs / surfeurs / ... peuvent s'en inspirer !

○ Les gouttelettes d'eau ruissellent sur une feuille de la fleur de lotus, qui est couverte de nombreuses pointes microscopiques qui empêchent l'eau de pénétrer.

- La nature a inspiré l'homme de multiples façons.
  - Les avions  $\Leftrightarrow$  structures d'ailes d'oiseaux.
  - Les robots et leurs mouvements des hommes  $\Leftrightarrow$  insectes.
  - Les matériaux résistants  $\Leftrightarrow$  toiles d'araignée.
  - Réseaux routier (logistique) imitant la nature (plantes), Algos Génés, &c.
- Les systèmes *biologiquement inspirés* ont pris de l'importance
  - ➔ beaucoup d'idées / innovations *imitent* la nature

**Constat** : certains systèmes sociaux dans la nature présentent un **comportement intelligent collectif**, même si elles sont composées de **simples individus** avec des capacités limitées (cf. les insectes).

☞ Voir support de ce cours séparé pour plus de détails.

- Les solutions intelligentes émergent naturellement de l'**auto-organisation** et de la **communication directe ou indirecte** entre des individus.
  - Ces solutions sont utilisées dans le développement de systèmes IA distribués.
  - **Stigmergie**  $\simeq$  (auto) incitation.
- **Notre propos** : le comportement collectif des fourmis
  - Leur capacité à trouver le plus court chemin entre leur nid et une source de nourriture
  - Imitée / utilisée pour résoudre des problèmes tels que TSP.
  - Plus généralement : la recherche de chemins dans un graphe, ...
- **Pour ce BE** : plusieurs sujets dont :

L'implantation d'un système **multi-agents** de recherche du chemin le plus court suivant le principe de la **Stigmergie** et utilisant les algorithmes **génétiques**.

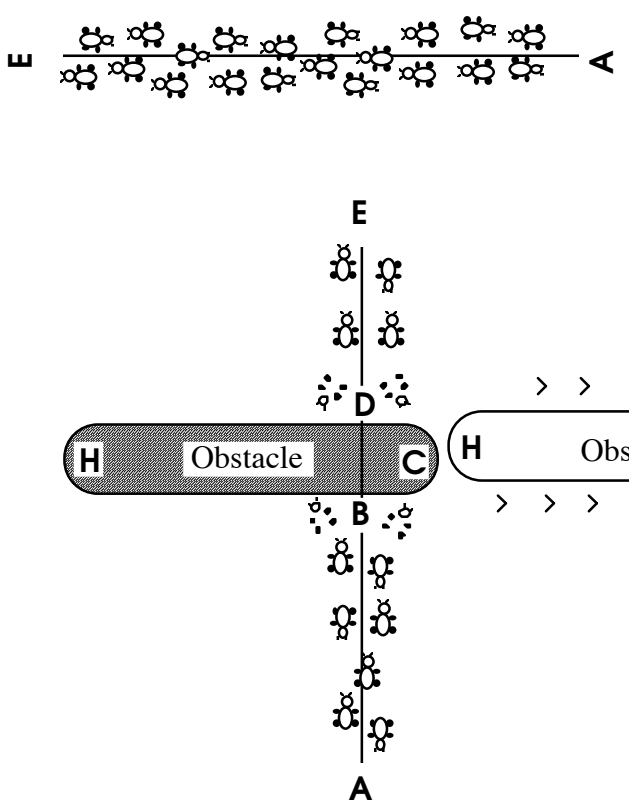
**On s'inscrit dans le cadre ACO** (*Ant Colony Optimization*) :

- L'optimisation de colonie de fourmis (ACO) étudie les systèmes artificiels qui imitent le comportement des vraies colonies de fourmis.
- Employée pour résoudre des problèmes discrets d'optimisation.
- Constitue une classe de méta-heuristiques pour des problèmes d'optimisation difficiles (NP-difficiles) et les problèmes **dynamiques** (en cours de résolution).
- Voir cours 2 pour "un peu d'histoire" sur ce sujet.

## I.1.1 Principes

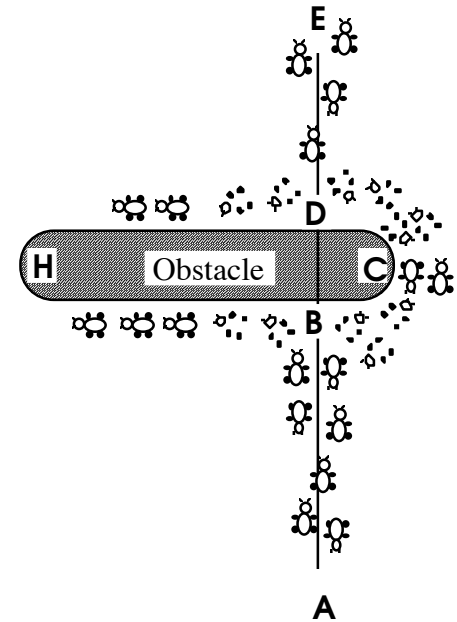
- "Fourmis" (artificielles) = Agents de recherche qui miment les comportement des fourmis réels.
- Question posée par les éthologues :
  - Comment des insectes qui ne voient pas trouvent le PCC entre leur nid et la nourriture ?
  - Réponse : utilisation du milieu (environnement) pour communiquer entre les individus.
- La **phéromone** : une fourmi qui se déplace laisse une quantité variable de phéromone sur son chemin, qui est détectée par les prochaines fourmis leur permettant de déterminer leur chemin avec une bonne probabilité.
- Une forme de **boucle rétroactive positive** ("autocatalytic behavior") :
  - **Plus les fourmis suivent un chemin, plus ce chemin devient attractif.**

## I.1.2 Exemple

- Les fourmis se déplacent entre le nid (E) et la source de la nourriture (A) sur le chemin A-E.
- 
- Un obstacle coupe le chemin.
  - La fourmi qui se déplace de A vers E et qui se trouve en B a 2 choix :
    - Le chemin B-C-D
    - Le chemin B-H-D
- $B-C-D$  ou  $B-H-D$ ? → le choix selon  $f(\text{intensité de la phéromone})$
  - La première fourmi a une même probabilité de suivre l'un de ces deux chemins.
  - Celle qui suit le chemin B-C-D arrive en premier en D ...

Puis (il faut retourner au nid) :

- Les fourmis qui retournent de E ont deux choix en D :
  - Le chemin D-C-B
  - Le chemin D-H-B
- Le chemin D-C-B aura une plus forte intensité de phéromone,
- Cette intensité est causée par :
  - La moitié des fourmis qui prennent ce chemin de retour.
  - Le nombre supérieur de fourmis qui auront suivi le chemin B-C-D et qui retournent au nid.
- Le chemin le plus court reçoit davantage de phéromone du fait du passage plus fréquent des fourmis.





### I.1.3 Phéromone : dépôt et évaporation

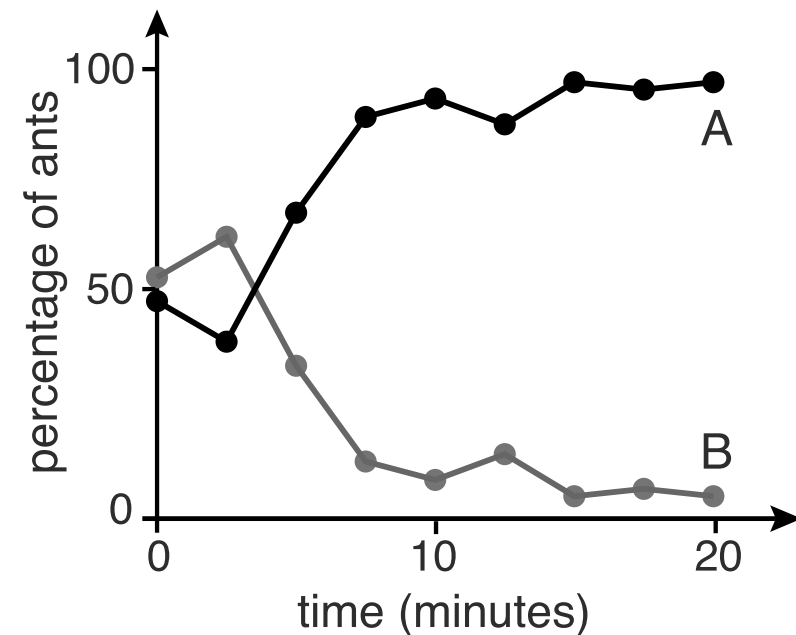
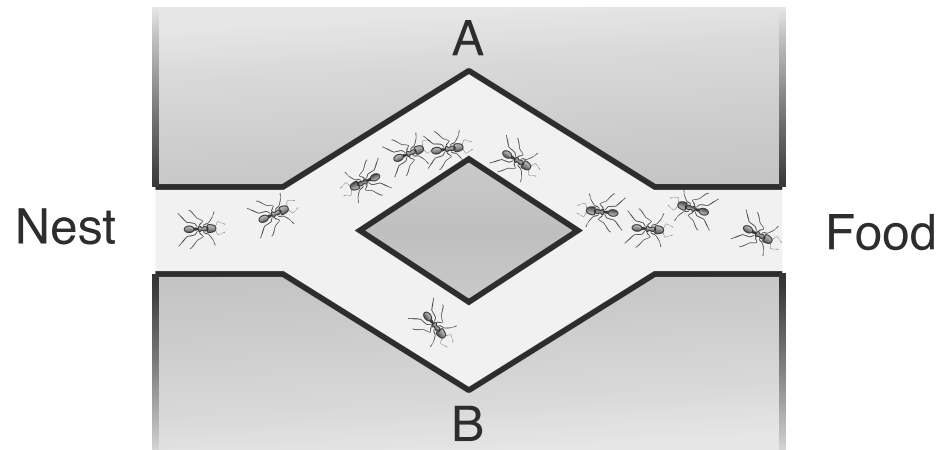


FIGURE I.1 – Dépôt et évaporation de Phéromone par les fourmis : La voie A saturée, la voie B quasi nulle

- ➔ On constate : moins une voie est empruntée, moins il y aura de phéromone dont la quantité tend vers 0 avec l'évaporation (dans le temps).
- ☞ Voir support de ce cours pour plus sur les détails du **rétrocontrôle positif**.

## I.2 Exemples d'application

### I.2.1 Essaim de robots collaborateurs (Projet Swarm-bots)

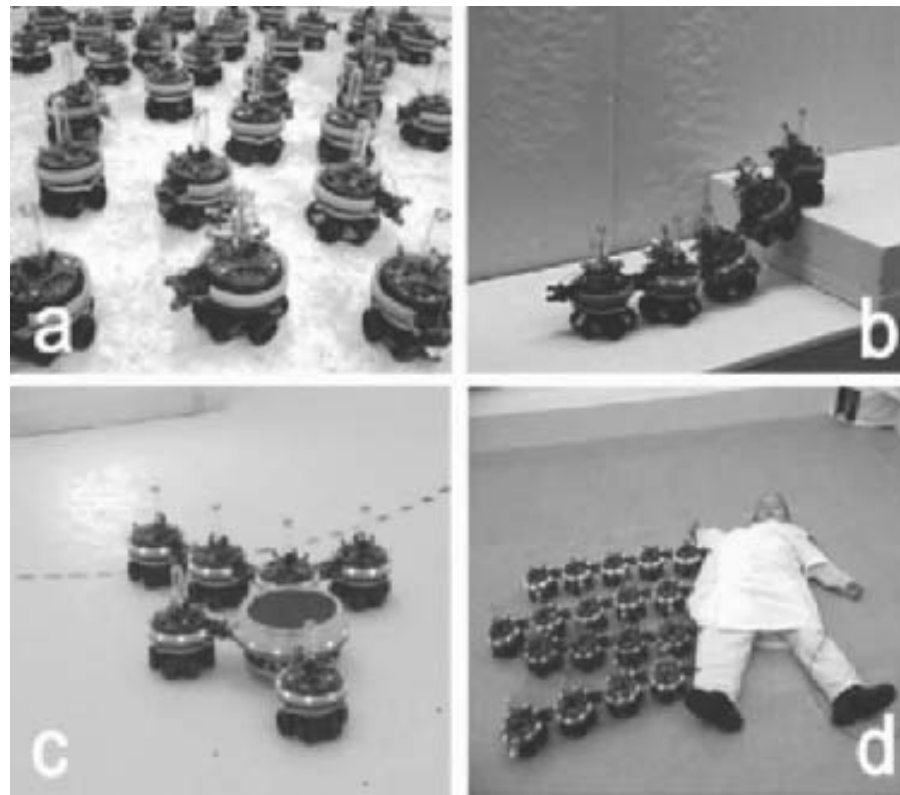


FIGURE I.2 – **Application** : projet swarm-bots : (<http://www.swarm-bots.org/>)

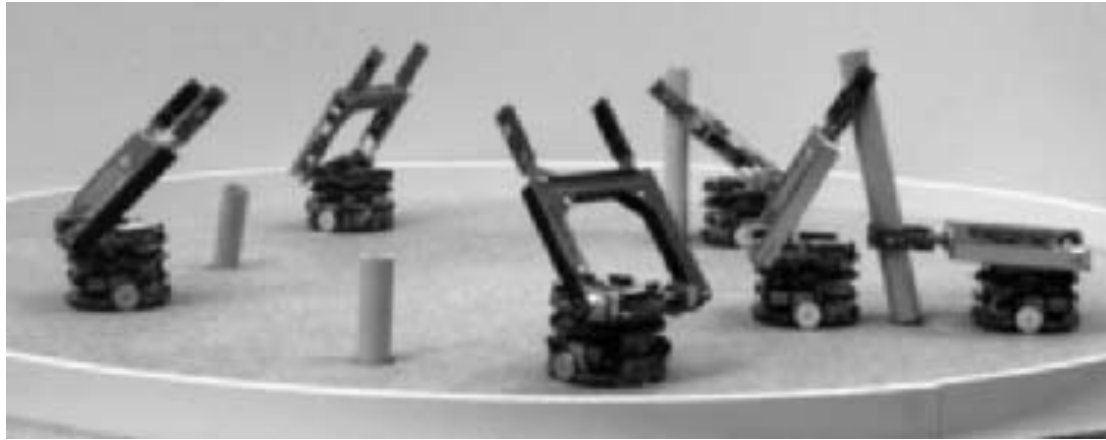


FIGURE I.3 – Une équipe de robots coopèrent pour tirer de longs bâtons sur le sol (Alcherio Martinoli, EPFL, Lausanne, Switzerland.)



FIGURE I.4 – projet swarm-bots : (<http://www.swarm-bots.org/>)

## I.3 Stigmergie, émergent

- Les métaphores biologiques et le modèle *stigmergique* ont été appliqués à la construction de nombreux systèmes complexes.
  - ☞ Tous ces systèmes utilisent des **agents** indépendants qui interagissent dans un environnement commun pour atteindre des propriétés globales.
- Le modèle de communication indirecte rend ces systèmes adaptatifs, décentralisée et émergentes.
- La **stigmergie** se caractérise par les éléments suivants :
  - Environnement "Open-hostile"
  - Pas de contrôle Centralisé
  - Pouvoir de calcul limité (convient à l'embarqué)
  - Communication élémentaire, peu (ou pas) de mémoire

## Par exemple, dans un réseau de communication :

- Ces caractéristiques sont souhaitables pour la survie et la sécurité du système.
- Être adaptatif et décentralisé est inhérent à la tolérance des pannes ;
- La décentralisation des moyens met le réseau davantage à l'abri d'attaques.
- Comportement "**émergent**" du système = le comportement global qui apparaît à partir des comportements des membres.
  - ➔ *Être émergent* permet d'éviter d'être l'unique élément problématique en cas de panne (comme dans les systèmes traditionnels).
- *A contrario*, les systèmes stigmergiques peuvent poser des problèmes de sécurité dès qu'ils fonctionnent dans un environnement ouvert et hostile.
  - ➔ Parce qu'il n'y a pas de contrôle centralisé (avec d'éventuels mécanismes cryptographiques coûteux mais parfois nécessaire) et seuls les membres possèdent une puissance de calcul limitée

## I.3.1 Optimisation par essaim de particules (PSO) : un exemple

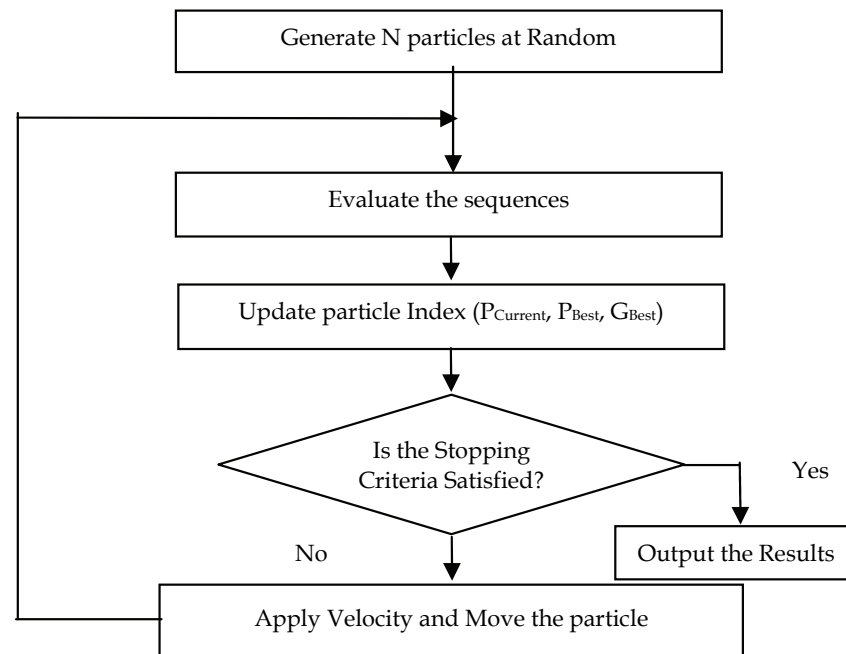


FIGURE I.5 – Algorithme de base de PSO

Après chaque "*déplacement*", les caractéristiques de la population (les particules) sont mises à jour via un calcul, par exemple par un code de "update" PSO suivant :

```

for each particle p
  for each coordinate c
    p.c = p.c + 2 * rand() * pbest.c - p.c + 2 * rand() * gbest.c - p.c
  
```

Où **pbest** : le meilleur local jusque là, **gbest** : le meilleur global jusque là.

→ P.ex : la fourmi la plus rapide au dernier voyage vs. la fourmi la plus rapide des tous les voyages.

☞ "centralisation" ?

## I.3.2 Quelques exemple d'application

- Applications au problème symétrique et asymétrique de voyageur de commerce.
- Applications au problème d'ordonnancement séquentiel.
- Applications au problème d'affectation quadratique.
- Applications au problème de tournées des véhicules.
- Applications aux problème d'établissement d'horaires (emploi du temps)
- Applications aux problèmes de coloration de graphe.
- Applications aux problèmes de partitionnement.
- Applications aux réseaux de télécommunications.
- Routage circuits, Implantations parallèles.
- &c.

**Dont des exemples d'applications** en Réseaux et Télécoms présentés dans la littérature :

- Network Routing Problems [DiCaro98]
- Peer-to-Peer network framework [Montresor01]
- Distributed Intrusion and Response Systems [Fenet01]
- Terrain Coverage [Koenig01],
- Modified PSO algorithm for solving planar graph coloring problem [Guangzha07]
- &c.

☞ Voir cours 2 : exemple de réseau (AntNet).



# I.4 BE : Modélisation de la colonie de fourmis comme un système multi-agents

- Cette section explique quelques éléments du BE.
  - On considère la recherche dans les graphes et le comportement alimentaire des fourmis
- Remarquer la similitude entre ces deux problèmes.

## I.4.1 Environnement : un graphe

- On considère les noeuds d'un graphe comme des lieux où les fourmis pourraient s'arrêter lors d'un "voyage" ;
  - On appellera ces noeuds "villes".
  - Les arêtes du graphe représentent les routes reliant ces villes.
- Cet **environnement** virtuel sera peuplé d'**agents** représentant les fourmis
  - *individus / population*
- Les fourmis essayent de trouver le meilleur itinéraire (le plus court chemin = PCC) entre deux points situés dans un environnement représenté par un graphe.
- L'idée principale : **mettre simplement des fourmis "en marche" sur un graphe** :
  - observer l'intensité de la phéromone déposée sur les arêtes du graphe dans le temps,
  - choisir des actions et **trouver le PCC**.

## I.4.2 Les agents

- En IA, un agent est une entité autonome qui interagit dans un (son) environnement.
- Les agents ont seulement une **perception** partielle mais dynamique de leur environnement qui peut être modifié/scruté par leurs capacités sensorielles  
→ capteurs en général, ici le dépôt de **phéromone**.
- Les agents peuvent effectuer des **actions** fondées sur la perception locale et sur leur représentation interne dans l'environnement.
- Les actions peuvent **affecter** l'environnement : l'agent lui-même ainsi que d'autres agents.
- On considère la description du comportement alimentaire des fourmis (avec le concept d'agent à l'esprit).

- Les agents auront besoin de :

- "marcher" (avancer) sur un **graphe**, d'une **ville** vers une autre ;
- "prendre" des aliments quand ils arrivent à la **source** de nourriture ;
- "laisser" cette nourriture lorsqu'ils reviennent au **nid**.
- "déposer" de la phéromone sur une **route** (une arête) lors d'un voyage ;
- "décider" / "choisir" quel sera le prochain chemin à suivre en fonction de l'intensité de phéromone sur les arêtes possibles depuis un noeud.

## Le principe de l'algorithme :

- Soit une route qui lie deux villes  $v_1$  et  $v_2$  :
    - On fera "avancer" les fourmis **pas à pas**  
= un pas d'itération fera avancer d'un pas chaque fourmi
    - Suite à ce "pas", on modifie l'état de l'agent (fourmi).
- ➔ Voir le tableau de la page suivante pour les actions.

- Le tableau suivant décrit les actions d'un agent en fonction de chaque Etat.

Lieu	Actions
Dans une ville (noeud)	Choisir l'arête suivante, déposer de la phéromone
Sur une route (arête)	Avancer un pas de plus
Au nid, transportant de la nourriture	Laissez les aliments
A la source de nourriture	Prendre de l'aliment et retourner au nid

- Un agent **sur un itinéraire** avance d'un pas (une étape) à chaque tour jusqu'à ce qu'il arrive dans une ville.
- Une fois **dans la ville**, il choisit un itinéraire en fonction de l'intensité de la phéromone sur les chemins disponibles.
- L'agent répète ce processus jusqu'à ce qu'il trouve la source de nourriture ; il prend la nourriture puis démarre **le voyage de retour** en suivant sa propre piste de phéromone
- De retour **dans le nid**, l'agent laisse la nourriture puis recommence le processus à nouveau.

☞ Notons qu'à chaque départ, les agents suivent les pistes ayant un maximum de Phéromone (**en départ, ils n'ont plus la mémoire de leur "propre" piste** ).

- Les biologistes pensent que la phéromone / la nourriture représente un **stimulus** (motivation).

☞ Notez que toutes ces actions nécessitent des informations locales et une mémoire à court terme permettant à l'agent de **reconnaître son propre chemin vers le nid**.

→ Les classes C++ nous aideront ! (hope)

- A propos du "retour par leur propre piste".

→ Évite à la fourmi de s'enfermer sur la même arête par ex. A-B (en faisant sans cesse A-B-A-B-A-...)

## I.5 La mise en oeuvre

- On simulera un environnement multi-agents en utilisant un système de tourniquet,
  - Comme un jeu où à **chaque tour**, tous les joueurs font un seul mouvement.
  - Chaque agent produira seulement l'une des actions élémentaires décrites ci-dessus à chaque tour.

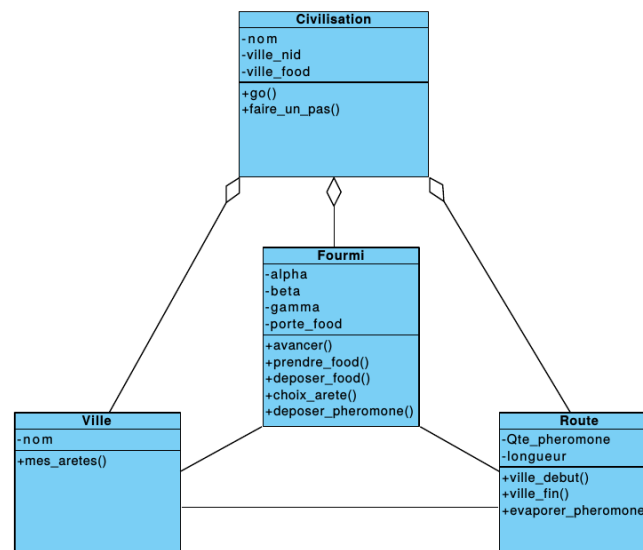


FIGURE I.6 – Une proposition possible (à compléter)

- Tous les éléments de l'environnement seront modélisés à l'aide classes (C++).
  - ➔ Les éléments à modéliser étant simples, la mise en oeuvre devient simple.
    - la classe **Ant** (fourmi) représente les agents,
    - la classe **Route** représente les arêtes du graphe,
    - la **Ville** représentent les noeuds et
    - la **Civilisation** représente l'environnement.
- La *ville* (noeud) est la plus simple entité :
  - ➔ On connaît sa position ( $X, Y$ ) dans l'environnement.
  - ➔ Nécessaire pour calculer la distance entre deux villes.
  - ➔ On peut représenter le "terrain" (l'environnement) par un échiquier.
- L'interface graphique de l'outil est traité a part.
  - ➔ Seuls les aspects essentiels seront abordés ici.



## I.5.1 La classe Route

- La route a besoin de deux pointeurs vers les deux villes qu'elle connecte.
- Une autre propriété est la longueur de la route (arête).
  - ➔ Plus une route est longue, plus l'agent a besoin de tours (des *pas*)
- L'intensité de phéromone sur une arête est représentée dans la classe Route.
- **Un point important** est la volatilité de la phéromone (à simuler).
  - ➔ La classe Route aura besoin d'une méthode pour simuler l'évaporation.

```
class Route
{
private:
    float longueur;           // Longueur de la route
    float pheromone;          // Intensite de la phéromone sur la route
    Ville *premiere;          // Villes connectées à cette route
    Ville *seconde;

public:
    void evaporer_Pheromone(); // Simulation de l'évaporation de la phéromone

    // constructeurs, interface et méthodes auxiliaires etc
};
```

### I.5.1.1 La classe Ant (agent, fourmi)

- Chaque instance de la classe **Ant** doit représenter un agent individuel avec des caractéristiques singulières.
- La caractéristique **la plus importante** d'une fourmi dans ce contexte est liée à sa tendance individuelle et imprévisible à choisir un itinéraire (arête) parmi ceux disponibles.
- Le niveau de phéromone sur une route est mesuré par un nombre réel.
- L'agent va utiliser une méthode qui évalue sa tendance à choisir un itinéraire en fonction de l'intensité de la phéromone.
  - Cette fonction évaluera la possibilité de choix de toutes les arrêts possibles ;
  - Puis choisira la **meilleure** parmi ces possibilités.

- Une bonne variabilité du comportement des agents peut être exprimée par une fonction sinusoïdale avec (au moins) trois coefficients  $\alpha, \beta, \gamma$ ,

$$PL_{t+1} = \alpha * \sin(\beta * PL_t + \gamma) \quad PL : \text{Pheromon Level sur la route}$$

- $\alpha, \beta$  et  $\gamma$  seront des propriétés de la classe **Ant** :
  - des réels aléatoires choisis dans l'intervalle  $[-5 \dots + 5]$ .
- D'autres paramètres possibles (pour les fourmis artificielles)
  - moduler le taux de phéromone "perceptible" sur l'arête,
  - la longueur de l'arête
  - l'"importance" de l'arête
- Ces propriétés permettront d'avoir des individus différentes dans la population
  - Elles sont également nécessaires pour les algorithmes génétiques qui seront abordés plus loin.

```
class Ant
{
private:
    float alfa;           // La sensibilité phéromonale de la fourmi
    float beta;
    float gamma;
    bool porte_nourriture; // Transporte de la nourriture ou non

public:
    float getTendance(int PheroLevel); // tendance à choisir une route
    void prendre_nourriture();         // Prendre la nourriture dans la source de nourriture
    void laisser_nourriture();         // Laisse la nourriture (dans le nid)
    void déposer_pheromone();          // Augmenter le niveau de la phéromone de la route
    void marcher();                    // Avancer une étape plus

    // constructeurs, interface et méthodes auxiliaires etc
};
```

### I.5.1.2 La classe Civilisation (Environnement)

- La classe *Civilisation* contrôle l'ensemble de l'environnement :  
→ le processus d'évolution et de simulation.
- Deux noeuds du graphe désignent le **nid** et la **source** de nourriture.
- Au début de la simulation, on crée un nombre aléatoire d'agents le nid.
- A chaque tour, les agents effectuent des actions en fonction de leur position actuelle.
- Pendant l'exécution de la simulation, les itinéraires les plus utilisés verront leur niveau de phéromone augmenter et après quelque temps, une solution émergera du comportement collectif de ces fourmis virtuelles.
- Si l'interface graphique (cf. Qt/Tk/etc.) est créée, un dessin du graphe peut être créé.

```
class Civilisation
{
private:
    Ville *src_nourriture;    // La ville source de nourriture de la Civilisation
    Ville *nid;              // Le nid de la Civilisation
    vector<Route*> routes;    // toutes les routes de l'Environnement
    vector<Ville*> villes;    // toutes les villes de l'Environnement
    vector<Ant*> fourmis;    // toutes les fourmis dans l'Environnement
    int selectionNaturelle; // les tours restants avant la prochaine sélection (pour l'algorithme génétique)
public:
    void tourSuivant();      // Effectue un tour de la simulation

    // constructeurs, interface et méthodes auxiliaires etc
};
```

→ On peut utiliser une liste à la place du vecteur.

## Remarques et critiques :

- Ce système peut donner de bons résultats,
  - Mais un nombre aléatoire d'agents ayant des caractéristiques aléatoires ne peut pas toujours résoudre un problème donné.
- Une population d'agents capable de trouver le PCC dans un graphe peut ne pas être en mesure de trouver une solution dans un environnement complètement différent.

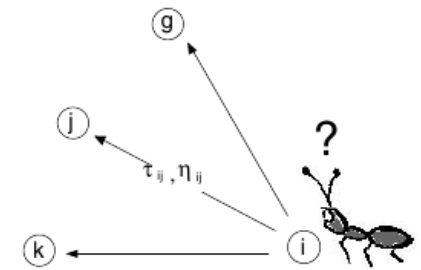
## D'où les questions :

- Comment trouver **les meilleurs agents** et combien en faut-il pour un problème donné ?
- Comment **équibrer** les agents ayant des caractéristiques différentes pour composer une bonne population ?
- **Amélioration** : vers les algorithmes **génétiques** discutés + loin (en optimisation)

# I.6 Meilleures règles sur la Phéromone

- On peut traduire les règles applicables au problème TSP avec l'optimisation "colonie de fourmis" (ACO) de différentes manières.

## Choix d'arête :



- Une fourmi décide de la prochaine ville en fonction du taux  $\tau_{ij}$  de phéromone (sur l'arête  $i - j$ ) et de l'heuristique  $\eta_{ij}$  associée à l'arête  $i - j$  (ici, la ville voisine  $j$  non encore visitée).
- La fourmi  $k$  dans la ville  $r$  choisit d'aller à la ville  $s$  ( $s$  : ville voisine de  $r$  non encore visitée par  $k$ , info. obtenue d'une mémoire locale  $M_k$  de la fourmi  $k$ ) via :

$$s = \begin{cases} \operatorname{argmax}_{s \notin M_k} \{ [\tau(r, s)] \cdot [\eta(r, s)]^\beta \} & \text{Si } q \leq q_0 \\ \operatorname{argmax}_{s \notin M_k} \{ P_k(r, s) \} & \text{Sinon} \end{cases}$$



**Détails** (voir  $P_k(r, s)$ )

- $\tau(r, s)$  est la quantité de phéromone sur l'arête  $(r, s)$
- $\eta(r, s)$  est une fonction heuristique associée à l'arête  $(r - s)$  qui est (souvent) l'inverse de la distance entre les villes  $r$  et  $s$ ,

$$s = \begin{cases} \operatorname{argmax}_{s \notin M_k} \{ [\tau(r, s)] \cdot [\eta(r, s)]^\beta \} & \text{Si } q \leq q_0 \\ \operatorname{argmax}_{s \notin M_k} \{ P_k(r, s) \} & \text{Sinon} \end{cases}$$

- $\beta$  est un paramètre qui pondère ici l'importance relative de la phéromone ainsi que la proximité ( $r \leftrightarrow s$ ),
- $q$  est un réel aléatoire (paramètre du système) choisi uniformément sur l'intervalle  $[0, 1]$ ,
- $0 \leq q_0 \leq 1$  est un paramètre du système,

- $P_k(r, s)$  est une probabilité de choix du prochain voisin  $s$  selon la distribution ci-contre (qui favorise les arêtes plus courtes avec un niveau de phéromone plus élevé)

$$P_k(r, s) = \begin{cases} \frac{[\tau(r, s)]^\alpha \cdot [\eta(r, s)]^\beta}{\sum_{s \notin M_k} [\tau(r, s)]^\alpha \cdot [\eta(r, s)]^\beta} & \text{Si } s \notin M_k, s \text{ voisine de } r \\ 0 & \text{Sinon} \end{cases}$$

☞  $P_k(r, s)$  est la probabilité de choisir d'aller de la ville  $r$  à  $s$ .

- Comme pour  $\beta$ , le paramètre  $\alpha$  pondère ici le taux de phéromone.

☞ On peut éviter de calculer  $P_k(r, s)$  : choisir la plus grande valeur du numérateur (sauf si on a explicitement besoin de  $P_k(r, s)$ , P. Ex.  $\{s | \text{Seuil}_1 \leq P_k(r, s) \leq \text{Seuil}_2\}$  où  $\text{Seuil}_i$  sont des paramètres du système).

☞ On peut simplifier et remplacer le calcul de  $P_k(r, s)$  par un tirage uniforme de  $s$  lorsque  $q \leq q_0$  (ci-haut).

☞ Voir ci-après : **une (3e) règle plus simple est proposée pour l'implantation** .

## I.6.1 Mise à jour de la phéromone

- MAJ lorsque les fourmis auront chacune construit leur trajet :
  1. D'abord **l'évaporation** : une baisse générale de la quantité sur **toutes les arêtes** du graphe par un facteur constant,
  2. Puis **augmentation** : on met à jour la phéromone sur les arêtes empruntées.

**La règle d'évaporation :**  $\tau'_{ij} \leftarrow (1 - \rho)\tau_{ij} \quad \forall (i, j) \in \text{graphe}$

Où  $0 \leq \rho \leq 1$  est le taux d'évaporation.

- $\rho$  permet d'éviter de stocker une quantité illimitée de phéromone et
- permet également de minimiser l'effet des "mauvais choix antérieurs".
- Une caractéristique de ces paramètres est qu'ils doivent permettre une évaporation *exponentielle* pendant les itérations si une arête n'est pas empruntée.

## La règle d'augmentation :

A la suite de l'évaporation, la quantité de phéromone sur les arêtes empruntées est augmentée :  $\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k \quad \forall (i,j) \in T^k$

Où  $\Delta\tau_{ij}^k$  est la quantité de phéromone que la fourmi  $k$  déposera sur l'arête empruntée.

Cette quantité est définie par :

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{C^k} & \text{Si l'arête } (i,j) \in T^k \\ 0 & \text{Sinon} \end{cases}$$

Où  $C^k$  est la longueur de la tournée  $T^k$  effectuée par la fourmi  $k$  ;

→  $C^k$  est simplement la somme des arêtes empruntées par la fourmi  $k$

## I.6.2 Initialisation des paramètres

- Au départ, on initialise les paramètres de la manière suivante :

$$\forall (i, j) \text{ arête du graphe, } \tau_{ij} = \tau_0 = \frac{m}{C^{nn}}$$

Où  $m$  est le nombre de fourmis et  $C^{nn}$  est la longueur de la tournée réalisée par l'heuristique "les plus proches voisins" (va simplement de proche en proche voisin).

☞ Ou bien par tout autre algorithme qui permet de construire raisonnablement un tour optimisé, au moins localement.

**N.B.** : Si la quantité initiale de phéromone  $\tau_0$  est trop basse, les résultats seront vite biaisés par la première tournée générée qui n'est en général pas une bonne solution.

→ Si  $\tau_0$  est trop élevée, on perdra beaucoup d'itérations jusqu'à l'évaporation des phéromones donnant des valeurs "raisonnables".

# I.7 Optimisation de la colonie de fourmis

- **Les algorithmes génétiques** : une technique d'optimisation adaptative basée sur le processus naturel d'évolution.
- Principe darwinien de la "survie du plus apte" au fil des générations consécutives.
- Les individus les plus efficaces propagent leurs caractéristiques par les gènes qui seront re-combinés dans la création de nouveaux individus.
- La population évolue en s'adaptant à leur environnement à travers la mutation et la sélection naturelle.
- Les agents sont créés au début avec des **caractéristiques aléatoires**.

## Remarques sur les caractéristiques :

- ils peuvent ne pas être adaptées (à un environnement spécifique)
- cela pourrait prendre trop de temps pour obtenir un bon résultat
- ils pourraient même ne pas trouver une solution.

## Les opérateurs évolutionnistes (génétiques) :

- **Sélection** : on retient les meilleurs acteurs (on élimine les autres),
- **Mutation** : souvent pour répondre à une nécessité (sinon à une perte d'individu),
- **Croisement** : progéniture prometteuse.
- Les performances du système peuvent être grandement améliorées en appliquant ces opérateurs à la population de fourmis.
- Deux types majeurs d'individus :
  - les **meilleurs travailleurs** (apportent beaucoup, en quantité)
  - les **meilleurs explorateurs** (les plus rapides ou les moins répétitifs).
  - Ajouter **vos critères** (fourmi meilleure en ?, fourmi qui perd la bouffe !, ...)
- ☞ Fourmis de force différente, fourmis soldats (des ennemies ?) ...

## I.7.1 Sélection

- La concurrence (compétition) : les individus retenus dans cet environnement peuvent
  - soit **recueillir** une grande quantité de **nourriture** (les *travailleurs*)
  - soit **trouver** des voies alternatives vers la source de nourriture (les *explorateurs*).
- La sélection naturelle leur permet de "produire" une descendance et de diffuser leurs caractéristiques.
  - De même que pour le processus naturel, l'évolution se produit parce que la nouvelle génération des individus sera parfois mieux que les parents.
- Extinction : les agents qui se **perdent** (dans l'envt.) ne peuvent pas/plus aider.
  - P. Ex. un agent qui voyage entre 2 noeuds en utilisant **toujours la même arête** est inutile à la colonie. De même, il sera **inutile s'il continue de tourner en rond**.
  - Ces individus seront éliminées de la population.

## I.7.2 Progéniture (croisement)

- **Croisement** (*crossover*) : les descendances sont créées sur la base des caractéristiques (pour nous,  $\alpha, \beta, \gamma$ ) des individus qui **réussissent**.
- Comme il y a **deux types d'individus nécessaires** à la colonie, deux individus seront créés à chaque cycle évolutif.
  - L'un d'eux sera descendant des deux travailleurs les plus réussis, et
  - L'autre sera descendant des deux meilleurs explorateurs.

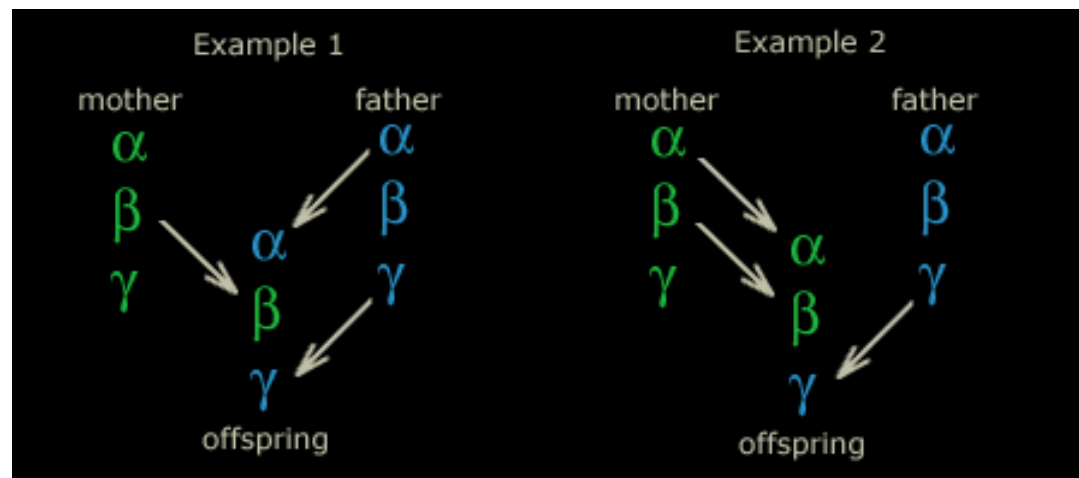


FIGURE I.7 – deux exemples de descendants créé par croisement



- Les gènes représentant les caractéristiques des parents seront combinés **au hasard** pour composer un nouveau chromosome qui donnera un nouvel individu.
- Cette combinaison est inspirée du processus biologique appelé **croisement**.
- Chaque caractéristique de l'individu viendra de l'un des parents choisi au hasard.
- La figure montre deux exemples possibles de descendants créé avec des combinaisons de croisement :

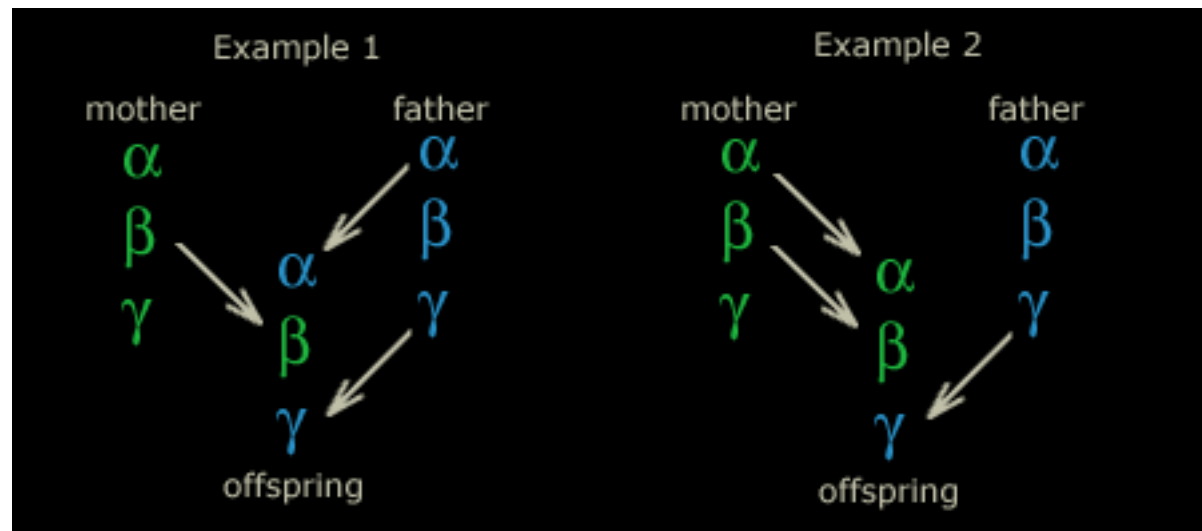


FIGURE I.8 – deux exemples de descendants créé par croisement

### I.7.3 Mutation

- La mutation peut maintenir la diversité au sein de la population.
- La **mutation** va changer l'une des caractéristiques de l'individu au hasard.
- ☞ Après un croisement, il y a une faible probabilité qu'une mutation se produise.

### I.7.4 Migration

La **migration** ajoutera un nouvel individu complètement aléatoire à la population.

- L'effet est similaire à la mutation, car elle va accroître la diversité au sein de l'environnement.

# I.8 Mise en oeuvre des opérateurs génétiques dans l'exemple

## I.8.1 Sélection

- Le **travailleur** le plus réussi est celui qui a recueilli davantage de nourriture.
- Un compteur sera ajoutée à la classe Ant et sera incrémenté à chaque fois que l'agent atteint le nid apportant de la nourriture.
- A chaque processus de sélection, l'agent avec la valeur supérieure de ce compteur sera considéré comme la plus réussi.
- Les agents comptent combien de fois ils ont été sur la même route (arête).
  - Les valeurs faibles de ce compteur indiquent les **explorateurs** avec succès ;
  - Les valeurs plus élevées : des agents qui se sont perdus dans l'environnement.

## I.8.2 Progéniture, Croisement

• **Croisement** (*crossover, recouvrement*) : un nouveau constructeur sera ajouté à la classe Ant qui aura pour paramètres deux références (pointeurs) sur *Ant*.

→ La nouvelle instance sera créée en combinant les caractéristiques des parents.

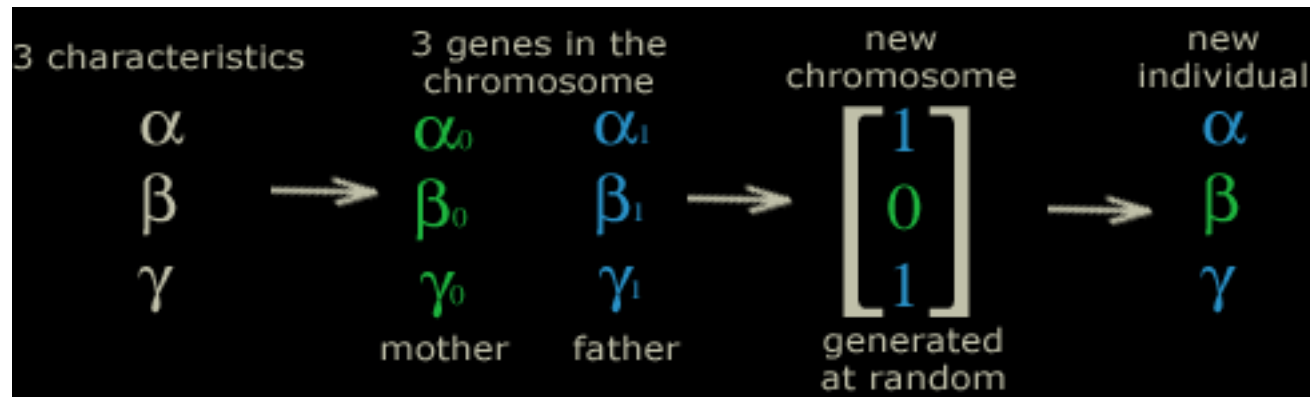


FIGURE I.9 – Implantation de CrossOver ( sur la base de  $\alpha, \beta, \gamma$  )

- Un chromosome est composé de 3 gènes représentant chacun une caractéristique.
- Le nouveau chromosome peut être rempli par des 0 ou des 1 avec la même probabilité.
- Pour chaque gène , 0 = la caractéristique sera héritée de la mère et 1 par le père.

### I.8.3 Mutation

- La **mutation** est une autre méthode qui sera utilisée pour effectuer une modification.
- Elle peut être appelée après le croisement (CrossOver) mais c'est un évènement avec une faible probabilité de se produire.

### I.8.4 Migration

- Un individu totalement nouveau peut être créé **en attribuant une valeur aléatoire** à  $\alpha$ ,  $\beta$  et  $\gamma$ .
  - Peut être mis en oeuvre comme le constructeur par défaut de la classe *Ant*.

## I.8.5 La nouvelle classe Ant

- Après l'inclusion de ces nouvelles propriétés et méthodes, la classe Ant devrait ressembler à ci-dessous (tous les attributs ne sont pas présents)

```
class Ant
{
private:
    float alfa;           // La sensibilité phéromonale de la fourmi
    float beta;
    float gamma;
    bool porte_nourriture; // Transporte de la nourriture ou non
    int qte_nourriture_collectee; // Quantité de nourriture collectée par cette fourmi
    int nb_fois_sur_meme_route; // voir ci-dessus

public:
    Ant(); // Constructeur par défaut : un individu tout neuf !
    Ant(Ant *papa, Ant *maman); // Constructeur Crossover
    float getTendance(int PheroLevel); // Evaluate la tendance à choisir une route (proba)
    void prendre_nourriture(); // Prendre la nourriture dans la source de nourriture
    void laisser_nourriture(); // Laisse la nourriture (dans le nid)
    void déposer_pheromone(); // Augmenter le niveau de la phéromone de la route
    void marcher(); // Avancer une étape plus
    void mutation(); // Faire une mutation

    // constructeurs, interface et méthodes auxiliaires etc ...
};
```

## I.9 Remarques

- Comme il n'y a pas de **fonction de coût explicite** liée au graphe, ce système peut être utilisé dans des applications avec un **environnement inconnus et temps réel** :
  - par exemple pour l'exploration robotique dans un environnement dynamique.
- On note qu'aucune connaissance préalable n'est nécessaire ici car l'effet d'une fonction de coût **implicite** est une conséquence de la longueur des chemins (arêtes) qui ont besoin de plus de temps pour être franchis.
- Les agents (fourmis) ont seulement une mémoire relativement limitée et le système n'est pas très affecté par la complexité du problème.
- Si l'un des agents est accidentellement perdu, le système continuera de travailler et le résultat final ne sera pas affectée.
  - Cette propriété rend le système **robuste** .

- La taille de la population est contrôlée par la sélection naturelle.
  - Si peu d'agents perdus → résultats plus rapides
  - Si des agents perdus (environnement vaste), ils seront éliminés.
  - La population va naturellement se développer dans des environnements plus vastes !
- Il n'y a pas de centre de décision central.
- L'information est distribuée entre les agents et l'environnement.
- Le système est également adaptatif.
  - Si une arête du graphe est soudainement enlevé ou même si une nouvelle est créée, le système peut rapidement se réadapter à l'environnement modifié et une nouvelle route va émerger.
- L'intelligence émergente peut être utilisée dans les jeux de stratégie temps réel.



- Les unités peuvent être conçues pour effectuer de simples actions locales avec un faible coût de calcul et de l'intelligence apparaîtraient alors comme une conséquence de l'auto-organisation de la colonie d'individus.
- Les algorithmes génétiques pourraient être appliqués pour sélectionner et faire évoluer les meilleurs individus et le joueur pourra apprendre de nouvelles stratégies et ainsi le jeu deviendrait plus intelligent.
- Ce système combine le principe bio-inspiré multi-agents, ceux des algorithmes génétiques et le principe de la *stimergie*.
- Le système est aisément **parallélisable**.

# I.10 D mo PCC

- PCC (simulation sous Windows, voir support du cours Chap2 pour les d tails)

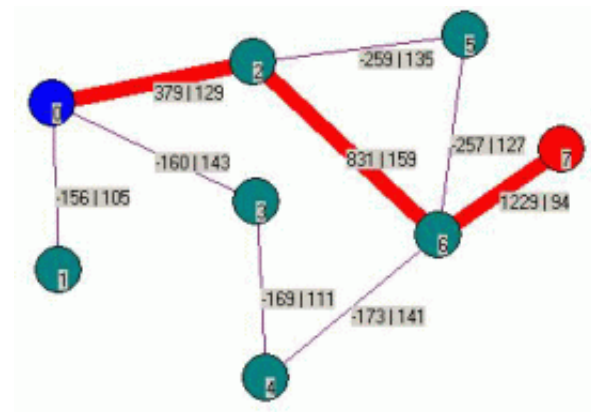
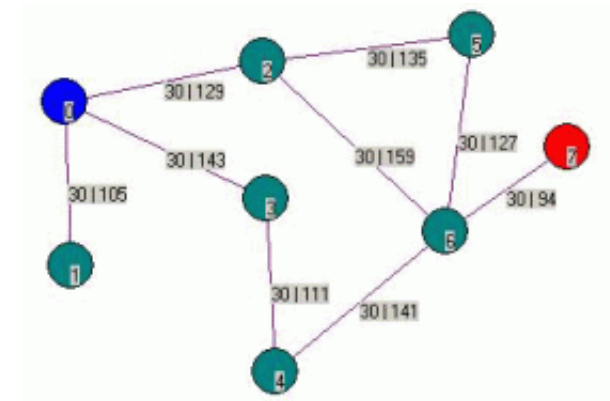
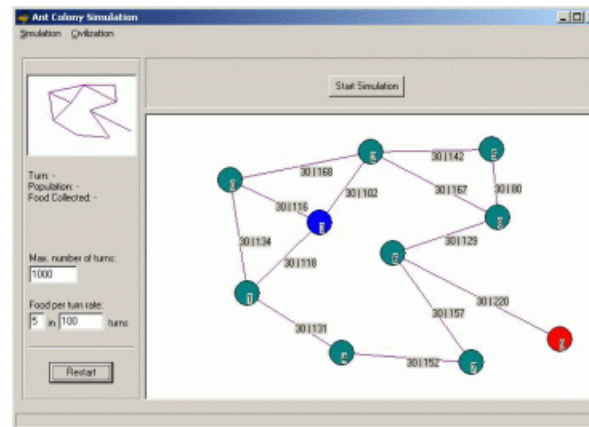


FIGURE I.10 – Meilleure route

# I.11 Sujet 2 : Coloration de graphes

**Coloration et colonie de fourmis** : 3 catégories de méthodes :

- 1) Chaque fourmi est un algo constructif qui laisse une trace sur chaque paire de sommets non adjacents pour indiquer si ces sommets ont reçu la même couleur.
- 2) Des fourmis se promènent sur le graphe et tentent collectivement de modifier la couleur des sommets qu'elles visitent, l'objectif étant de diminuer le nombre d'arêtes conflictuelles dans une k-coloration non acceptable.
- 3) Les fourmis sont devenues des algorithmes de recherche locale qui laissent des traces sur l'exploration qu'elles ont faite de l'espace de recherche.
  - Les algorithmes les plus récents (ceux de la troisième catégorie) rivalisent positivement avec les meilleurs algorithmes connus à ce jour.
  - Voir "mise en oeuvre" ci-dessous.

### I.11.1 Mise en oeuvre (proche de la 3e)

- Soit un graphe de  $N$  noeuds et un ensemble de  $k$  couleurs,  $k \leq N$ .

La 1e étape des calculs suivants a lieu **en parallèle** au niveau de chaque noeud.

☞ Il n'y a pas de notion de distance et toutes les fourmis envoyées arrivent en même temps à destination

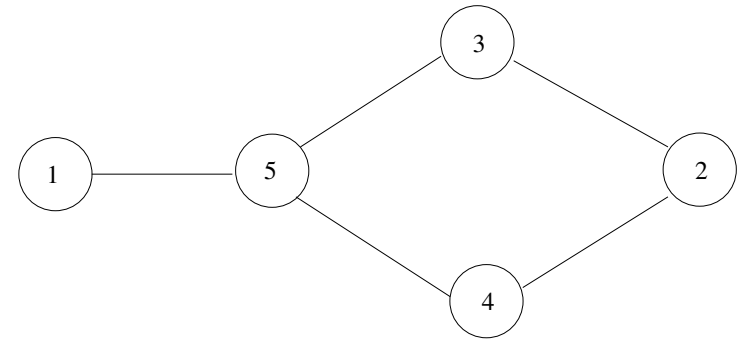
- Chaque noeud a un degré  $D$  de connexions et lâchera  $D$  fourmis locales, une par voisin.
- Au départ, on active chaque noeud en parallèle avec les autres.
- Chaque noeud choisit une couleur  $C$  et envoie une fourmi par voisin pour lui **dire de ne pas prendre  $C$** .

Ce message peut être traduit par le dépôt de phéromone sur les **autres** couleurs ( $\neq C$ ) du voisin : éliminer une couleur chez le voisin = favoriser les **autres** couleurs :        .. / ..

- Le messenger porte le numéro de l'envoyeur.
- L'action du choix d'une couleur au niveau d'un noeud peut être confiée à une fourmi supplémentaire par noeud.
- **Itération** : une décision locale est prise au niveau de chaque noeud  $M$  :
  - Examiner les fourmis présentes dans  $M$ .
    - ➔ Chacune est là pour dire : "ne prend pas telle couleur" (i.e. elle favorise les autres couleurs).
    - ➔ Ordonner les noeuds selon le nombre maximum d'un même message venant des voisins différents (voir exemple ci-dessous).
  - Choisir le noeud qui maximise ce nombre (venant d'un même noeud, le dernier message annule le précédent)
  - Lui donner une couleur non contestée.
  - Refaire l'itération

## I.11.2 Exemple

- Soit ce graphe avec 5 noeuds et un ensemble de  $k \leq 5$  couleurs  $\{R, V, B, \dots\}$
- Au départ, chacun choisit la couleur  $R$  (rouge) et envoie autant de fourmis que de connexions à ses voisins.



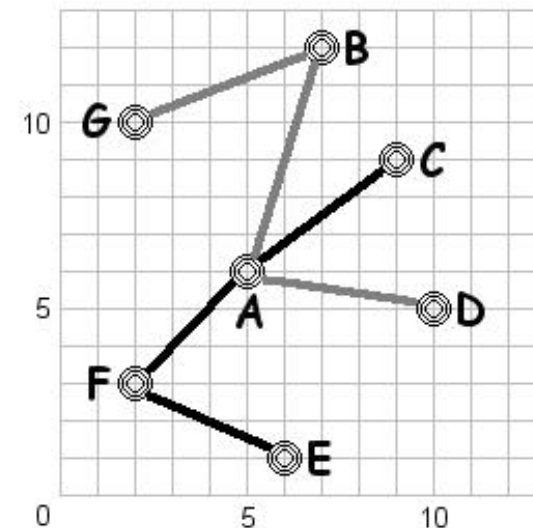
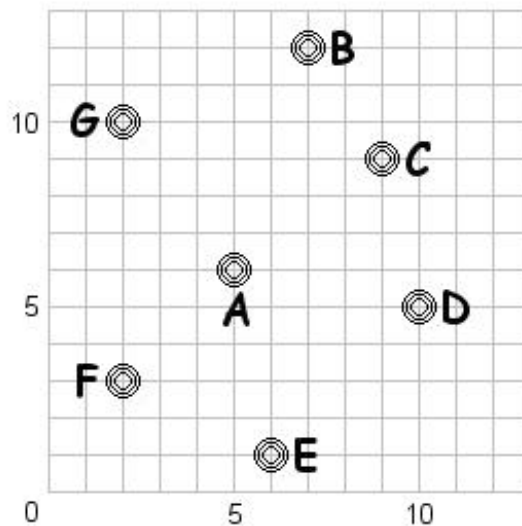
- La table suivante résumé les actions (avec optimisation mais non parallèle) :  
../..

Commentaires et décisions	1	2	3	4	5
1er tour, tous choisissent $R$ Et envoient le message $\bar{R}_{dest}$ aux voisins	$R$ $\bar{R}_5$	$R$ $\bar{R}_{3,4}$	$R$ $\bar{R}_{2,5}$	$R$ $\bar{R}_{2,5}$	$R$ $\bar{R}_{1,3,4}$
Comptage ( <b>maxi</b> même $\bar{C}$ de $\neq dest$ ) → Choix de 5 (ne prend pas $R$ ) Et envoi de 3 fourmis $\bar{V}_5$ aux voisins	1 $\bar{V}_5$	2	2 $\bar{V}_5$	2 $\bar{V}_5$	<b>3</b> prend <b>V</b>
Etat messages (nb. fourmis présentes + messages)	$\bar{R}_5, \bar{V}_5$	$\bar{R}_{3,4}$	$\bar{R}_{2,5}, \bar{V}_5$	$\bar{R}_{2,5}, \bar{V}_5$	<b>V</b>
Annulation du message $\bar{R}_5$ (5 a pris V) → Venant de 5, le nouv. messenger $\bar{V}_5$ tue le préc $\bar{R}_5$	$\bar{V}_5$	$\bar{R}_{3,4}$	$\bar{R}_2, \bar{V}_5$	$\bar{R}_2, \bar{V}_5$	<b>V</b>
Max même couleur exp. différents ☞ un nouv. message du même exp. annule le préd. Choix de 2 (qui ne prend pas $R$ ) Et envoi de fourmis $\bar{V}_2$ aux voisins	1	<b>2</b>  <b>V</b>	1  $\bar{V}_2$	1  $\bar{V}_2$	
<b>Bilan</b> : ( $\bar{V}_2/\bar{V}_5$ annule $\bar{R}_2 / \bar{R}_5$ car 2 et 5 ont pris V) Les 3 noeuds restants sont déconnectés et prennent R	$\bar{V}_{2,5}$	<b>V</b>	$\bar{V}_{2,5}$	$\bar{V}_{2,5}$	<b>V</b>

☞ **Optimisation** : on peut calculer en parallèle 2 noeuds non connectés.

# I.12 Sujet 3 : Bus Allocation Problem (BAP)

- Ce problème se compose d'un certain nombre d'arrêts et de lignes de bus.  
→ Chaque arrêt de bus est à un endroit précis.
- La figure suivante donne une telle carte avec 7 arrêts ; la solution est à droite.



- Utilisant l'emplacement de chaque arrêt de bus, nous sommes en mesure de calculer la distance euclidienne entre deux arrêts.



- Notez qu'il est également possible d'utiliser des routes prédéfinies à la place de la distance euclidienne.
- Cela pourrait signifier que la distance entre deux arrêts X à Y est différente de la distance de Y à X.
- Un voyage d'un arrêt (départ) à un autre (destination) est appelé une **transition**. Nous supposons que pour chaque transition possible, le nombre de passagers est connu.
- Un des arrêts de bus sera l'arrêt principal. Cette arrêt représentera la **gare centrale**.
- Dans le modèle suivant, toute ligne de bus devra faire appel à l'arrêt principal.
- Une solution à la BAP sera une collection de lignes de bus.
- L'arrêt principal sera associé à l'ensemble des lignes de bus mais tous les autres arrêts seront affectés à exactement une ligne de bus.
- Les bus circuleront simultanément du première arrêt au dernier et du dernier au premier.

## Mise en oeuvre :

- On utilisera :

$D = \{d_{ij}\}$  = distance entre deux arrêts  $i$  et  $j$ .

$T = \{t_{ij}\}$  = nbr personnes en attente à l'arrêt  $i$  ayant l'arrêt  $j$  pour destination.

- On note également :

$r_m$  : arrêt principal

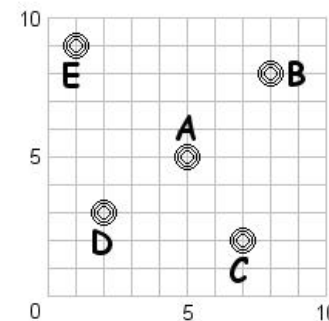
$v$  : vitesse moyenne du bus (m/s)

$f$  : fréquence du bus (temps/h).

$c$  : le temps nécessaire pour changer de bus (correspondance).

- Pour simplifier, on se donne l'instance plus simple suivante avec sa matrice  $T$  des transitions

→ Rappel :  $T$  : nbr personnes en attente à l'arrêt  $i$  ayant l'arrêt  $j$  pour destination.



$$T = \begin{bmatrix} 0 & 7 & 10 & 10 & 7 \\ 5 & 0 & 3 & 3 & 1 \\ 9 & 4 & 0 & 7 & 5 \\ 9 & 3 & 6 & 0 & 4 \\ 7 & 4 & 4 & 3 & 0 \end{bmatrix}$$

- En utilisant une solution, la vitesse moyenne du bus et la matrice  $D$ , on peut calculer la matrice suivante :

$U = \{u_{ij}\}$  = Le temps nécessaire pour une personne d'aller de l'arrêt  $i$  à  $j$ .

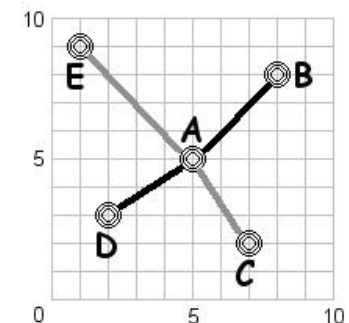
- Pour comparer les solutions, on se donne également la valeur *att* pour "average travel time" qui devra être minimisée (fonction objective) via

$$att = \frac{\sum_{x=1}^n \left( \frac{\sum_{y=1}^n u_{xy} \cdot t_{xy}}{\sum_{y=1}^n t_{yz}} \right)}{n}$$

- Soit la matrice des distances  $D$  pour l'instance simplifiée, sa matrice  $U$  et une solution à cet exemple :

$$D = \begin{bmatrix} 0 & 424.26 & 360.56 & 360.56 & 565.69 \\ 424.26 & 0 & 608.28 & 781.03 & 707.11 \\ 360.56 & 608.28 & 0 & 509.90 & 921.95 \\ 360.56 & 781.03 & 509.90 & 0 & 608.28 \\ 565.69 & 707.11 & 921.95 & 608.28 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & 112.117 & 132.32 & 112.117 & 132.32 \\ 60.6092 & 0 & 132.32 & 112.117 & 372.32 \\ 51.5079 & 352.117 & 0 & 352.117 & 132.32 \\ 51.5079 & 112.117 & 132.32 & 0 & 372.32 \\ 80.8122 & 352.117 & 132.32 & 352.117 & 0 \end{bmatrix}$$

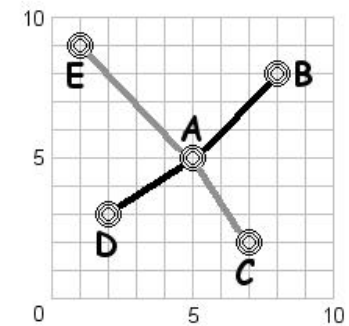


../..

## Explications de la matrice $U$ :

$$D = \begin{bmatrix} 0 & 424.26 & 360.56 & 360.56 & 565.69 \\ 424.26 & 0 & 608.28 & 781.03 & 707.11 \\ 360.56 & 608.28 & 0 & 509.90 & 921.95 \\ 360.56 & 781.03 & 509.90 & 0 & 608.28 \\ 565.69 & 707.11 & 921.95 & 608.28 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & 112.117 & 132.32 & 112.117 & 132.32 \\ 60.6092 & 0 & 132.32 & 112.117 & 372.32 \\ 51.5079 & 352.117 & 0 & 352.117 & 132.32 \\ 51.5079 & 112.117 & 132.32 & 0 & 372.32 \\ 80.8122 & 352.117 & 132.32 & 352.117 & 0 \end{bmatrix}$$



$u_{21}$  : (B à A) : B est alloué à la même ligne de bus que A, et cette entrée est égale à la valeur du  $d_{21}$  (distance de B à A) divisé par 7 (vitesse du bus).

$u_{12}$  : (de A à B) : A est alloué à la même ligne de bus que B, mais la personne qui attend A en premier doit attendre le bus qui arrive de D avant de pouvoir aller à B : cet entrée est donc est égale à  $u_{42}$  (de D à B).

$u_{43}$  : (D à C) : D n'est pas attribué à la même ligne de bus que C.

- Une personne doit d'abord se rendre à A, où elle découvre qu'elle peut encore prendre le bus car :

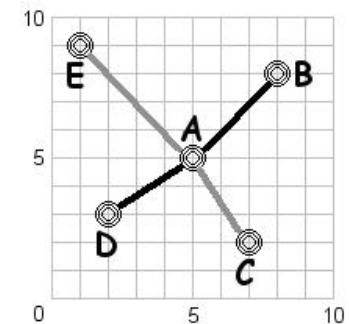
- de E à A prend 80,81 secondes et D à A prend 51,51 secondes.

- Donc  $51,51 + 20$  secondes (ce qui est  $c$ ) est toujours inférieur à 80,81 secondes.

- Quand elle arrivera à C, cela lui aura pris  $80.81 + 51.51$  (A à C) secondes.
- $u_{54}$  : (E à D) : E n'est pas attribué à la même ligne que D.
- Encore une fois la personne se déplace d'abord à A, mais elle découvre qu'elle a raté le bus et doit attendre le prochain.
- Quand elle arrive à D, il lui aura pris sa 240 secondes (somme de la 1e colonne de la matrice  $U$  en tenant compte de la fréquence  $(3600 / f)$  pour les attentes).
- Utilisant la matrice  $U$ , la matrice et l'équation de "*att*", nous sommes en mesure de calculer *att*. La valeur de *att* de cette solution est 155,491 secondes.
- Donc, il faut en moyenne à une personne 155,491 secondes pour arriver à destination.

$$D = \begin{bmatrix} 0 & 424.26 & 360.56 & 360.56 & 565.69 \\ 424.26 & 0 & 608.28 & 781.03 & 707.11 \\ 360.56 & 608.28 & 0 & 509.90 & 921.95 \\ 360.56 & 781.03 & 509.90 & 0 & 608.28 \\ 565.69 & 707.11 & 921.95 & 608.28 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & 112.117 & 132.32 & 112.117 & 132.32 \\ 60.6092 & 0 & 132.32 & 112.117 & 372.32 \\ 51.5079 & 352.117 & 0 & 352.117 & 132.32 \\ 51.5079 & 112.117 & 132.32 & 0 & 372.32 \\ 80.8122 & 352.117 & 132.32 & 352.117 & 0 \end{bmatrix}$$



# I.13 Sujet 4 : Coloration par PSO

- Lire et implanter l'article

"Modified PSO algorithm for solving planar graph coloring problem"

déposé à coté de ce support.

# I.14 Quelques References

1. Marais, E. N., de Kok, W.(trans.) 1937 : The Soul of the White Ant [http ://journeytoforever.org/farm\\_library/Marais1/whiteantToC.html](http://journeytoforever.org/farm_library/Marais1/whiteantToC.html)
2. Maeterlinck, Maurice. 1927 : The Life of the White Ant. Allen & Unwin
3. Grassé, P.-P. 1959 : La Reconstruction du nid et les coordinations interindividuelles. La théorie de la stigmergie, Insectes Sociaux 6 : 41-84.
4. Goss. S., Aron. S., Deneubourg, J.L. and Pasteels, J.M. 1989 : Self-organized shortcuts in the Argentine ant. Naturwissenschaften 76, 579-581.
5. Benzatti, D. 2002 : Emergent Intelligence, AI Depot [http ://ai-depot.com/CollectiveIntelligence/Ant.html](http://ai-depot.com/CollectiveIntelligence/Ant.html)
6. Dorigo, M. and Gambardella, L.M.. Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1(1) :53-66, 1997. [http ://citeseer.nj.nec.com/article/dorigo96ant.html](http://citeseer.nj.nec.com/article/dorigo96ant.html)
7. Schoonderwoerd, R., Holland, O., Bruten, J. and Rothkrantz, L., 1996 : Ant-based load balancing in telecommunications networks, Adaptive Behavior, vol.5, No.2, .
8. Di Caro, G and Dorigo, M. 1998 : AntNet : Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research, 9 :317-365, . [http ://citeseer.nj.nec.com/dicaro98antnet.html](http://citeseer.nj.nec.com/dicaro98antnet.html)
9. Eberhart, R. C., and Kennedy, J. 1995 : A New Optimizer Using Particles Swarm Theory, Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43.
10. Collective Robotic Intelligence Project (CRIP) - [http ://www.cs.ualberta.ca/~kuba/research.html](http://www.cs.ualberta.ca/~kuba/research.html)
11. [http ://www.swarm-bots.org](http://www.swarm-bots.org)
12. J. Hoffmeyer, The swarming body, Proc. 5th Congress of the International Association for Semiotic Studies, Berkeley, 1994. [http ://www.molbio.ku.dk/MolBioPa](http://www.molbio.ku.dk/MolBioPa)

# Table des matières

I.1	Intelligence Emergente	1
I.1.1	Principes	5
I.1.2	Exemple	6
I.1.3	Phéromone : dépôt et évaporation	8
I.2	Exemples d'application	9
I.2.1	Essaim de robots collaborateurs (Projet Swarm-bots)	9
I.3	Stigmergie, émergent	11
I.3.1	Optimisation par essaim de particules (PSO) : un exemple	13
I.3.2	Quelques exemple d'application	14
I.4	BE : Modélisation de la colonie de fourmis comme un système multi-agents	16
I.4.1	Environnement : un graphe	17
I.4.2	Les agents	18
I.5	La mise en oeuvre	22
I.5.1	La classe Route	24
I.5.1.1	La classe Ant (agent, fourmi)	25
I.5.1.2	La classe Civilisation (Environnement)	28
I.6	Meilleures règles sur la Phéromone	31
I.6.1	Mise à jour de la phéromone	33
I.6.2	Initialisation des paramètres	35
I.7	Optimisation de la colonie de fourmis	36



I.7.1	Sélection	38
I.7.2	Progéniture (croisement)	39
I.7.3	Mutation	41
I.7.4	Migration	41
I.8	Mise en oeuvre des opérateurs génétiques dans l'exemple	42
I.8.1	Sélection	42
I.8.2	Progéniture, Croisement	43
I.8.3	Mutation	44
I.8.4	Migration	44
I.8.5	La nouvelle classe Ant	45
I.9	Remarques	46
I.10	Démo PCC	49
I.11	Sujet 2 : Coloration de graphes	50
I.11.1	Mise en oeuvre (proche de la 3e)	51
I.11.2	Exemple	53
I.12	Sujet 3 : Bus Allocation Problem (BAP)	55
I.13	Sujet 4 : Coloration par PSO	61
I.14	Quelques References	62