Problème du voyageur de commerce

ING1 – Projet de Parcours MI

Année 2013–2014

EISTI

Modalités

— Ce projet durera 30h et sera effectué par des **groupes de 2 (ou 3)** personnes.

— La note finale comprendra : la réalisation, le rapport et la soutenance.

— Le rendu de la réalisation sera au plus tard le **jeudi 12 juin**, à **23h59**.

— La soutenance aura lieu le **vendredi 13 juin**.

— Le rendu du rapport sera au plus tard le **vendredi 13 juin**, à **23h59**.

— *Rappel : un rapport est composé d'une page de garde, d'un sommaire, d'un résumé et d'un abstract*[1]*, d'une bibliographie, d'un glossaire (si besoin), en plus du contenu du rapport.*

— Le rapport sera rendu au format pdf, et la réalisation seront rendus au format tgz (ou tar.gz).

— La présence est **obligatoire (9h-12h30 et 14h-17h30). Une pénalité individuelle de -2 par demi-journée d'absence sera appliqué.**

1 Introduction

Problème du voyageur de commerce

> "Dans le problème du voyageur de commerce, qui est très proche du problème du cycle hamiltonien, un représentant doit visiter n villes. En modélisant le problème par un graphe complet à n sommets, on peut dire que le représentant souhaite faire une tournée, ou un cycle hamiltonien, en visitant chaque ville exactement une fois, et en terminant sa tournée dans la ville de départ. Le voyage entre la ville i et la ville j a un coût entier $c(i, j)$, et le représentant souhaite effectuer la tournée dont le coût total est minimal, le coût total étant la somme des coûts individuels le long de chaque arête de la tournée." [CLRS04]

1. résumé en anglais

Problématique du projet

Résoudre exhaustivement le problème du voyageur de commerce n'est praticable qu'avec un nombre très réduit de villes. Afin de pouvoir traiter les cas plus complexes, nous allons nous tourner vers une heuristique, c'est-à-dire une méthode de calcul qui fournit une solution approchée de l'optimum global. Dans ce projet, nous utiliserons l'optimisation par essaims particulaires (ou *particle swarm optimization (PSO)*) .

2 Heuristique : PSO

L'idée initiale des essaims particulaires [PKB07] est de se baser sur les interactions sociales entre particules plutôt que sur la capacité cognitive de chaque individu. Ce modèle collaboratif s'est inspiré de certains comportements animaliers comme le vol des oiseaux, les bancs de poissons, ...

Le principe de base du PSO est le suivant : un ensemble de particules est initialement disposé dans l'ensemble de recherche. Chaque particule est composée et connait :

— une position et une vitesse

— sa meilleure position

— ses voisins [2], avec leurs meilleures positions

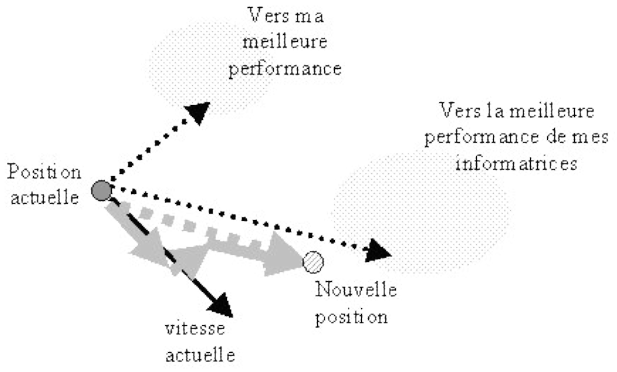

FIG. 1. Particule en déplacement

À chaque étape, le comportement de la particule est un compromis entre trois choix possibles (Figure 1) :

— suivre son propre chemin

2. On considèrera que la particule fait partie de son voisinage

— aller vers sa meilleure ancienne position

— aller vers la meilleure position de son voisinage

Le compromis est formalisé par l'équation suivante :

$$\begin{cases} v_{t+1} &= c_1 v_t + c_2(x_{best} - x_t) + c_3(x_{vois} - x_t) \\ x_{t+1} &= x_t + v_{t+1} \end{cases} \tag{1}$$

avec :

— v_t : la vitesse à l'instant t

— x_t : la position à l'instant t

— x_{best} : la meilleure ancienne position, à l'instant t

— x_{vois} : la meilleure position du meilleur voisin, à l'instant t

— c_1, c_2, c_3 : coefficient d'inertie et de confiance

À chaque itération, les particules vont se déplacer afin de trouver une solution optimale approchée du problème. Le schéma de la figure 2 correspond aux différentes étapes de l'algorithme PSO.

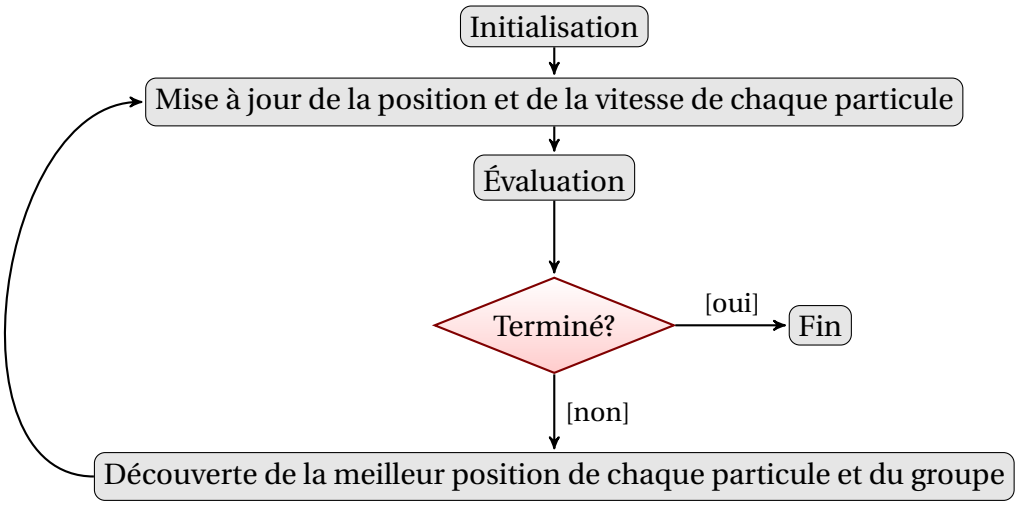

FIG. 2. Algorithme PSO

3 PSO appliqué au problème du voyageur de commerce

Pour appliquer l'algorithme de PSO au problème du voyageur de commerce, il suffit de déterminer à quoi correspondent la position, la vitesse, ainsi que les différentes opérations de base (addition, soustraction, produit).

Position : Ceci correspond à une solution de la particule, donc dans notre cas à un chemin possible. Par exemple : $(1,3,4,0,2,5)$

Vitesse : C'est ce qui permet de faire déplacer une particule, donc de changer la position. Comme la position correspond à un chemin possible, ce qui nous permet de passer d'un chemin à un autre est un ensemble de permutations. Par exemple : $\{[1,3],[4,2],[0,3]\}$

Addition (Position + Vitesse = Position) : Appliquer une vitesse à une position correspond à appliquer les permutations à un chemin.

Soustraction (Position - Position = Vitesse) : La différence entre deux positions correspond à une vitesse. Donc la différence entre deux chemins correspond à un ensemble de permutations permettant de passer d'un chemin à l'autre.

Produit (constante * Vitesse = Vitesse) : Ce produit permet de pondérer la vitesse à appliquer à la particule. Dans notre cas où la vitesse correspond à un ensemble de permutations, nous avons à faire à plusieurs cas de figure. Notons c la constante et v la vitesse.

- $c = 0$: aucune permutation n'est gardée
- $c \in]0,1]$: on tronque les n premières permutations, où $n = \lceil c * \|v\| \rceil$, où $\|v\|$ correspond au nombre de premutations
- $c > 1$: on peut écrire $c = k + c'$, $k \in \mathbb{N}$, $c' \in [0,1[$ d'où $c * v = (k + c') * v = k * v + c' * v = v + ... + v + c'v$

4 Réalisation

Lors de ce projet, vous devez résoudre le problème du voyageur de commerce en utilisant l'heuristique PSO. Vous trouverez sur le site de TSPLIB (`http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`) un ensemble de jeux de test, ainsi que quelques optimums connus. Nous utiliserons les données symétriques. Les données peuvent être sous deux formats différents :

xml : Le graphe est directement donné sous forme de sommets et d'arêtes valuées. Vous pouvez utiliser la librairie Jdom pour parser votre fichier xml.

tsp : Le graphe est donné sous forme de sommets dont on connait la position. Le graphe est complet, et la valuation des arêtes est à calculer en tant que distance euclidienne.

Afin de faciliter l'accès à l'information, il est conseillé de garder le graphe sous forme de matrice d'adjacence.

L'algorithme de PSO décrit au dessus contient un certain nombre de paramètres à fixer :

Nombre de particules : on en choisit généralement entre 20 et 50.

Constante $c1$: on prend généralement une valeur dans $]0,1[$.

Constantes $c2$ et $c3$: elles sont tirés aléatoirement dans $]0,2[$ à chaque itération.

Nombre d'itérations : assez grand pour que les particules aient le temps de parcourir l'espace de recherche.

Références

[CLRS04] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction à l'algorithmique*. Dunod, 2004.

[PKB07] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.