L'objectif de ce projet est de résoudre le problème de voyageur de commerce (TSP) par l'algorithme discret PSO.

Un voyageur de commerce doit visiter un certain nombre de villes une et une seule fois.

Étant données des distances entre chaque paire de villes, on doit minimiser la distance totale parcourue.

Le TSP peut être représenté par un graphe : chaque ville correspond à un sommet et chaque arête à une paire de villes pouvant être visitées l'une à la suite de l'autre

Le TSP consiste à trouver un tour complet (circuit Hamiltonien) dans ce graphe qui minimise la somme des distances.

Le Nombre de solutions possibles pour N « villes » : **N! / 2.**

Les essaims particulaires (Particle Swarm Optimization PSO)

Cette métaheuristique s'appuie notamment sur un modèle développé par le biologiste Craig Reynolds à la fin des années 1980, permettant de simuler le déplacement d'un groupe d'oiseaux.
Cette méthode d'optimisation se base sur la collaboration des particules (des solutions possible)entreelles. Elle a d'ailleurs des similarités avec les algorithmes de colonies de fourmis, qui s'appuient eux aussi sur le concept d'auto-organisation.

Ainsi, grâce à des règles de déplacement très simples (dans l'espace de solutions), les particules peuvent converger progressivement vers un optimum.

Au départ de l'algorithme, un essaim est réparti au hasard dans l'espace de recherche de dimension D (nombre de villes), chaque particule p est aléatoirement placée dans la position x^p de l'espace de recherche, chaque particule possède également une vitesse aléatoire. Ensuite, à chaque pas de temps :

– chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance P^i(dans notre cas le meilleur ordre des villes visitées) : la meilleure position qu'elle a atteinte jusqu'ici (qui peut en fait être parfois la position courante) et sa qualité (la valeur en cette position de la fonction à optimiser),

– chaque particule est capable d'interroger un certain nombre de ses congénères (ses informatrices, dont elle-même) et d'obtenir de chacune d'entre elles sa propre meilleure performance P^g ,

– à chaque pas de temps, chaque particule choisit la meilleure des meilleures performances dont elle a connaissance, modifie sa vitesse V en fonction de cette information et de ses propres données et se déplace en conséquence.

La modification de la vitesse est une simple combinaison linéaire de trois tendances, à l'aide de coéfficients de confiance :

– la tendance « aventureuse », consistant à continuer selon la vitesse actuelle,
– la tendance « conservatrice », ramenant plus ou moins vers la meilleure position déjà trouvée,
– la tendance « panurgienne », orientant approximativement vers la meilleure informatrice.
La mise à jour des deux vecteurs vitesse et position, de chaque particule p dans l'essaim, est donnée par les équations (1) et (2) :

$$v_j^p(t+1) = \tau(t)v_j^p(t) + \mu\omega_{1j}(t)(P_j^i(t) - x_j^p(t)) + \nu\omega_{2j}(t)(P_j^g(t) - x_j^p(t)) \quad (1)$$

$$x_j^p(t+1) = x_j^p(t) + v_j^p(t+1) \quad (2)$$

Ou j= 1,..,D (D est le nombre de villes), $\tau(t)$est le facteur d'inertie, μ représente le paramètre cognitif et ν le paramètre social. ω_{1j}(t) et ω_{2j}(t) sont des nombres aléatoires compris entre 0 et 1.

Principales caractéristiques de l'algorithme PSO

Ce modèle présente quelques propriétés intéressantes, qui en font un bon outil pour de nombreux problèmes d'optimisation, particulièrement les problèmes fortement non linéaires, continus ou mixtes (certaines variables étant réelles et d'autres entières) :

– il est facile à programmer, quelques lignes de code suffisent dans n'importe quel langage évolué,

– il est robuste (de mauvais choix de paramètres dégradent les performances, mais n'empêchent pas d'obtenir une solution).

Algorithme Pseudo code de l'algorithme PSO

$t \leftarrow 0$

Pour chaque particule // solution possible

Initialiser sa position et sa vitesse.

Initialiser $P^i(t)$

Fin pour

Répéter

Choisir la particule $P^g(t)$ ayant la meilleure fitness dans l'itération courante

Pour chaque particule p

Calculer la vitesse $v^p(t + 1)$ utilisant l'équation (1).

Mettre à jour le vecteur position $x^p(t + 1)$ selon l'équation (2).

Calculer la valeur de la fitness $f(x^p(t))$

Si $f(x^p(t)) > f(P^{i(}t))$

$P^i(t + 1) \leftarrow x^p(t)$

Fin si

Fin pour

$t \leftarrow t + 1$

Tant que (critère d'arrêt n'est pas validé)

Implémentation de l'algorithme d'optimisation par essaims particulaires à la résolution de problème de voyageur de commerce (PVC):

Le problème de voyageur de commerce (PVC) opère uniquement sur des villes, les villes peuvent être représentées par des entiers, chaque ville est représentée par un chiffre. La représentation d'une solution de PVC se compose d'une séquence ordonnée de nombres entiers. De ce fait, Le but de ce projet consiste en l'application de l'algorithme d'optimisation discrète par essaims particulaires pour la résolution de FS-FAP.

PSO a été à l'origine conçu pour traiter des problèmes dont l'espace de recherche est réel multidimensionnel. Pour pouvoir appliquer PSO à la résolution de PVC, le modèle utilisé doit être adapté à ce type de représentation de solution (c-à-d. la position d'une particule est maintenant une séquence ordonnée de nombres entiers).

Le codage utilisé nécessite la redéfinition des éléments (position et vitesse) et des opérations (multiplication externe d'un coefficient par une vitesse, la somme de vitesses et la somme d'une vitesse et une position) des équations (1) et (2). En plus, il est nécessaire de déterminer comment la population initiale doit être choisie et de définir les critères d'arrêt les plus adéquats.

- **Position de la particule :** Une position se compose d'une solution qui représente la séquence des villes visitées. Chaque membre de l'essaim se compose de D paramètres entiers, où D est le nombre de villes. Par exemple, une position peut être une séquence (1, 6, 4, 2,5, 3), s'il s'agit de 6 villes.

- **Vitesse de la particule :** L'expression *X*2-*X*1 où *X*1 et *X*2 sont deux positions, représente la différence entre deux positions et la vitesse demandée pour aller de *X*1 à *X*2. Cette vitesse est une liste ordonnée de transformations (appelées mouvements) qui doivent être appliquées séquentiellement à la particule pour passer de sa position courante *X*1 en une autre *X*2. Un mouvement est une paire de valeurs α/j.
Pour chaque position u dans la séquence *X*1, l'algorithme détermine si l'unité qui est en position u de la séquence *X*1 est la même unité qui est en position u de la séquence *X*2. Si les unités sont différentes, *fi* est l'unité en position u de *X*2 et j est égal à la position u. Ainsi, ce mouvement dénote que pour aller de la séquence *X*1 à la séquence *X*2, l'unité en position j doit être échangée par l'unité.α Par exemple soient *X*1 = (*A*1*; B*1*; C*2*; C*1*; B*2*; C*4*; A*2*; C*3) et *X*2 = (*A*1*; C*1*; B*2*; C*2*; A*2*; C*4*; B*1*; C*3), donc la vitesse *X*2 -*X*1 est constituée par la liste de mouvement : *{*(*C*1/2)*;* (*B*2/3)*;* (*C*2/4)*;* (*A*2/5)*;* (*B*1/7)*}.*

$$
\begin{aligned}
X_1 &= (A_1, B_1, C_2, C_1, B_2, C_4, A_2, C_3) \\
(C_1/2) &\rightarrow (A_1, C_1, C_2, B_1, B_2, C_4, A_2, C_3) \\
(B_2/3) &\rightarrow (A_1, C_1, B_2, B_1, C_2, C_4, A_2, C_3) \\
(C_2/4) &\rightarrow (A_1, C_1, B_2, C_2, B_1, C_4, A_2, C_3) \\
(A_2/5) &\rightarrow (A_1, C_1, B_2, C_2, A_2, C_4, B_1, C_3) \\
(B_1/7) &\rightarrow (A_1, C_1, B_2, C_2, A_2, C_4, B_1, C_3) = X_2
\end{aligned}
$$

Multiplication externe d'un coefficient par une vitesse :

Les valeurs des coefficients des $\tau(t)$ et ν utilisés dans l'équation de la mise à jour du vecteur vitesse (équation (1)) sont entre 0 et 1. Lorsqu'un coefficient est multiplié par une vitesse, il indique la probabilité de chaque mouvement à être appliquer.

Par exemple, si on multiplie le coefficient 0.6 par la vitesse [(*C*1/2)*;* (*B*2/3)*;* (*C*2=4)*;* (*A*2/5)*;* (*B*1/7)], cinq nombres aléatoires entre 0 et 1 sont génerés pour la comparaison avec la valeur 0.6. Si le nombre aléatoire est inférieur à 0.6, le mouvement est appliqué. Par conséquent, si les valeurs des nombres aléatoires sont 0*:*8*;* 0*:*3*;* 0*:*7*;* 0*:*4 et 0*:*2, les mouvements (*B*2=3)*;* (*A*2=5) et(*B*1=7) sont appliqués, tandis que les mouvements (*C*1/2) et (*C*2/4) ne sont pas appliqués. La vitesse résultante de la multiplication est donc [(*B*2=3)*;* (*A*2=5)*;* (*B*1=7)], qui, comme précédemment indiqué, représente une liste de mouvements qu'on va appliquer à un point.

Somme de vitesses : La somme de deux vitesses est simplement la concaténation de leur propre liste de mouvements.

Somme de vitesse et une position : La somme d'une vitesse et une position donne le résultat d'appliquer séquentiellement chaque mouvement de la vitesse à la position.

Population initiale : La population initiale est produite en plaçant une vitesse vide et une position aléatoire pour chaque particule.

-Critère d'arrêt : L'algorithme s'arrête après avoir effectuer un nombre pré-défini d'itérations (40 pour notre cas).

Problème :

On suppose qu'on une séquence de ville V1, V2,V3,V4 ,V5, V6 et on cherche à visiter chaque ville toute en minimisant la distance parcourue.

Distance entre ville	V1	V2	V3	V4	V5	V6
V1	0	70	60	300	140	100
V2	70	0	90	260	100	50
V3	60	90	0	240	60	170
V4	300	260	240	0	100	200
V5	140	100	60	100	0	100
V6	100	50	170	200	100	0

Travaille demandé :

Appliquer l'algorithme DPSO à une population de 40 individus évoluant durant 60 générations.