Recherche Tabou

On étudiera la recherche Tabou sur le problème du voyageur de commerce avec 10 et 50 villes (problème du plus court chemin).

I. Introduction

1. Copiez le fichier TP_RechercheTabou.zip sur votre compte local à partir Moodle (AG41, https://moodle.utbm.fr/) et procédez à sa décompression.
2. Chargez le programme dans un environnement de programmation C++, compilez (en tapant '`make`' en ligne de commande ou en utilisant Code::Blocks par exemple) et exécutez (en tapant '`algo_tabou.exe`' en ligne de commande sans argument ou avec 4 arguments:'`algo_tabou.exe nb_itérations durée_tabou nb_villes distances_villes`').

II. Présentation du squelette du programme

3 classes (rechercheTabou et solution et random avec des fichiers .h et .cpp), une fonction main.cpp, 2 fichiers de distances et un fichier 'make'

1. La fonction main()
2. La classe solution
   a. le constructeur solution()
   b. la fonction evaluer()
3. La classe rechercheTabou
   a. le constructeur RechercheTabou()
   b. la fonction voisinage_2_opt()
   c. la fonction optimiser()

III. Analyse de paramétrage

A. Nombre d'itérations avec durée Tabou nulle (problème à 10 villes)

1. Lancez l'algorithme d'optimisation plusieurs fois avec les paramètres suivants :
   Nombre d'itérations : 5, 10, 100.
   Durée tabou : 0
   Nombre de ville : 10
   Nom du fichier : distances_entre_villes_10.txt
2. Au bout de combien d'itérations l'algorithme converge ? Après que fait-il ? Quelles conclusions en tirez-vous ?

***Remarque* : pour le problème de 10 villes le minimum global est 3473.**

3. Calculer le nombre de solutions au problème du voyageur de commerce avec 10 villes et 50 villes.
4. Calculer le nombre de voisins (au sens de voisinage 2-opt) d'une solution du problème du voyageur de commerce avec 10 villes et 50 villes.
5. Combien de solutions l'algorithme visite il avant de converger ? Que peut-on dire de la performance de l'algorithme ?

B. Nombre d'itérations avec durée Tabou nulle (problème à 50 villes)

1. Lancez l'algorithme d'optimisation plusieurs fois avec les paramètres suivants :
   Durée tabou : 0
   Nombre de ville : 50
   Nom du fichier : distances_entre_villes_50.txt
   Nombre d'itérations : 10, 100, 1000.

2. Au bout de combien d'itérations l'algorithme converge ? Après que fait-il ? Quelles conclusions en tirez-vous ?
3. Combien de solutions l'algorithme visite il avant de converger ? Que peut-on dire de la performance de l'algorithme ?

***Remarque* : pour le problème de 50 villes la fitness minimale trouvée est 5644.**

C. Durée Tabou (1ième partie)

1. Faites intervenir maintenant la liste Tabou. Lancez l'algorithme d'optimisation plusieurs fois avec les paramètres suivants :
   Nombre d'itérations : 1000.
   Durée tabou : 10
   Nombre de ville : 50
   Nom du fichier : distances_entre_villes_50.txt
2. Comment de fois évolue la meilleure solution trouvée ?
3. Combien de minimums locaux l'algorithme visite-t-il ?
   L'algorithme converge-t-il ?
   Quelles conclusions en tirez-vous ?

D. Durée Tabou (2ième partie)

1. Comment est codée la liste Tabou ? Qu'est ce qui est tabou ? Comment est codée la durée Tabou ? Quelle est donc la durée Tabou maximum ?

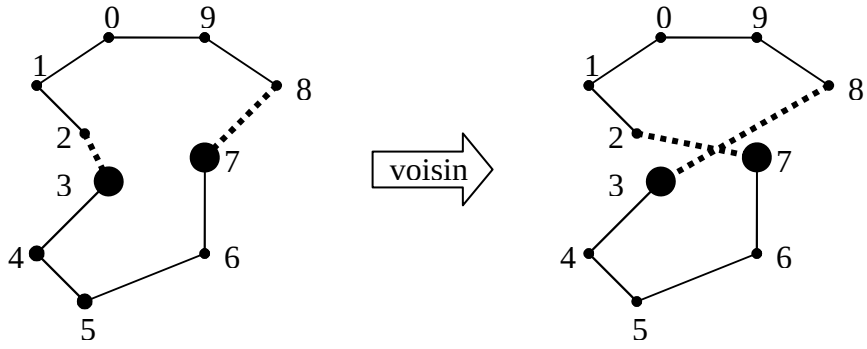
2. Relancez l'algorithme d'optimisation plusieurs fois (5 fois pour chaque valeur des paramètres) en gardant les mêmes paramètres sauf la durée Tabou :
   Durée tabou : 20, 50, 80, 100 et 1000.
3. Comment de fois évolue la meilleure solution trouvée ?
   Combien de minimums locaux l'algorithme visite-t-il ?
   L'algorithme converge-t-il ?
   Quelles conclusions en tirez-vous ?
4. Quel paramétrage (nombre d'itération, durée Tabou) proposez vous ?

5. Quelles conclusions en tirez-vous et quel serait l'impact d'une valeur de durée tabou approchant $+\infty$?

IV. Développement du critère d'aspiration

Proposez et implémentez un critère d'aspiration pour améliorer les performances de l'algorithme.

Annexe : Le voisinage 2-opt



0-1-2-***3***-**4**-**5**-**6**-***7***-8-9 0-1-2-***7***-**6**-**5**-**4**-***3***-8-9