IFT2015 - Travail Pratique #2

Ce travail pratique consiste en l'implantation d'un algorithme donnant des solutions approximatives au problème du voyageur de commerce. Ce problème consiste à trouver le plus court chemin qui visite chaque noeud d'un graphe exactement une fois et termine au noeud de départ. Dans ce travail, nous nous concentrons sur les graphes non-orientés complets où l'inégalité triangulaire est respectée.

Une heuristique vorace, nommée la méthode du plus proche voisin, consiste à commencer à un noeud aléatoire et de voyager vers le voisin le plus proche non visité jusqu'à-ce que chaque noeud le soit. Cette heuristique donnera des résultats différents dépendamment du point de départ. Soit L_{nn} le coût du chemin minimal donné par cette méthode appliquée à tous les points de départ possibles. En moyenne, cette heuristique donne un chemin 25% plus long que le chemin optimal, quoi qu'il soit techniquement possible qu'elle nous donne la solution avec le coût le plus élevé dans le pire des cas.

Une méthode approximative supérieure proposée par Dorigo et Gambardella en 1996 utilise une colonie de fourmis artificielles pour découvrir des bonnes solutions. Ce modèle est inspiré du comportement des vraies fourmis qui peuvent trouver les chemins les plus courts entre leur nid et les sources de nourriture en suivant les pistes de phéromones déposées par les autres fourmis.

Cet algorithme est décrit comme suit: chaque arête est initialement donnée un niveau de phéromones $\tau_0 = (n \cdot L_{nn})^{-1}$ où n est le nombre de sommets. Ensuite, m fourmis sont déposées sur des noeuds de départ aléatoires. À chaque itération, les fourmis voyagent vers un nouveau noeud non-visité. Les fourmis doivent donc enregistrer les noeuds visités dans leur mémoire M_k où k représente la $k^{\text{ème}}$ fourmi. À chaque fois qu'une fourmi visite un noeud, elle met à jour le niveau de phéromones de l'arête utilisée pour rejoindre ce noeud selon la règle de **piste locale**:

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \tau_0$$

où $\tau(r,s)$ est le niveau de phéromones sur l'arête entre les noeuds r et s. Une fois que chaque fourmi a trouvé une solution, la fourmi ayant la solution la moins coûteuse rebrousse son chemin en mettant à jour le niveau de phéromones sur **chaque arête composant sa solution** selon la règle de **piste globale**:

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s), \text{ où } \Delta\tau(r,s) = (\text{Coût de la solution})^{-1}$$

La mise à jour des phéromones selon la règle de piste globale est la dernière opération de l'itération. À la prochaine itération, le tout est recommencé alors que les niveaux de phéromones sont conservés (les fourmis sont déposées sur des nouveaux noeuds aléatoires, etc.). Après plusieurs itérations, les phéromones guident les fourmis vers de bonnes solutions.

La politique de décision qui détermine le noeud s qu'une fourmi doit visiter à partir du noeud r est la suivante:

$$s = \begin{cases} \arg\max\limits_{u \notin M_k} \{[\tau(r,u)] \cdot [\eta(r,u)]^\beta\} & \text{Si } q \leq q_0 \\ S & \text{Sinon} \end{cases}$$

où $\eta(r,u)$ est l'inverse de la distance entre les noeuds r et u, β est un paramètre déterminant l'importance d'un noeud proche par rapport aux niveaux de phéromones, q est une valeur tirée aléatoirement d'une distribution uniforme ayant comme intervalle $[0,1]$, q_0 est un paramètre donnant la probabilité de prendre une décision déterministe et S est une variable aléatoire sélectionnée selon la distribution suivante:

$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)] \cdot [\eta(r,s)]^\beta}{\sum\limits_{u \notin M_k} [\tau(r,u)] \cdot [\eta(r,u)]^\beta} & \text{Si } s \notin M_k \\ 0 & \text{Sinon} \end{cases}$$

Après chaque itération, vous devez imprimer le chemin de la fourmi qui met à jour la piste globale ainsi que le coût de ce chemin. La sortie de votre programme devrait être similaire à `result.txt` et votre implantation devrait converger vers la solution optimale pour le problème `dantzig.csv` qui a un coût de 699.

Bonus

Pour "résoudre" le plus gros problème `d198.csv`, il est possible de maintenir une liste de candidats tel qu'expliqué par l'article de Dorigo et Gambardella. Vous devez implanter cette optimisation, trouver une solution ayant un coût inférieur à 16000 et remettre un fichier nommé `bonus.txt` qui est la sortie de l'exécution du programme qui a trouvé cette solution.

Remise: 3 Juillet 2022, 23h55

Performance	70%
Élégance et clarté du code	30%
Bonus	10%
Total	110%