Complexité
Travaux Dirigés n° 2
Complexité

Nicolas Monmarché

10 octobre 2016

1 Reconnaissance de séquences

Étant données une séquence α et une grammaire $G = (\mathcal{A}, \mathcal{V}, S, \mathcal{R})$, proposez un algorithme permettant de savoir si α fait partie du langage engendré par G (autrement dit : $\alpha \in L(G)$?).

2 SAT

1. Étant donnée une expression logique sous forme normale conjonctive, proposez un algorithme permettant de générer toutes les affectations des variables propositionnelles.
2. Étant donnée une expression logique sous forme normale conjonctive et une affectation des variables propositionnelles, proposez un algorithme qui détermine si cette affectation est satisfaisable. Par exemple, avec :

$$(p \vee \neg q) \wedge p \wedge (q \vee \neg r \vee s \vee \neg t) \wedge (s \vee t)$$

et $(p, q, r, s, t) = (1, 1, 0, 1, 0)$, l'expression logique est-t-elle vraie ?

3 3-SAT

Le problème 3-SAT est similaire à SAT à la différence que les expressions logiques sont composées de clauses formées de 3 littéraux exactement. Par exemple, l'expression logique suivante a cette propriété :

$$(p \vee \neg q \vee \neg r) \wedge (p \vee s \vee q) \wedge (q \vee s \vee \neg t) \wedge (s \vee t \vee \neg p)$$

1. Montrer que le problème 3-SAT est dans NPC.

4 TSP par la pratique

Nous allons nous intéresser à la complexité du problème du voyageur de commerce (TSP). Le programme `tsp.c` reçoit en argument (sur la ligne de commande) un nombre de villes à

utiliser, génère ces n villes à des positions aléatoires puis évalue (c'est-à-dire calcule la longueur d'un chemin) de toutes les solutions possibles et retourne la meilleure trouvée. Nous utiliserons ce programme pour évaluer la complexité du TSP.

1. Ecrire la fonction `Permutation` qui teste toutes les permutations des n villes pour découvrir la meilleure solution :

   ```
   double permutation(TCoord * c, int n, int * best_sol)
   ```

2. En utilisant le programme `tsp.c`, tracer la courbe $C_1(n, t)$ où n représente le nombre de villes et t la durée nécessaire à l'obtention de l'optimum.
3. Tracer la courbe C_2 sur le même principe que C_1 mais en désactivant tous les affichages (en particulier les permutations évaluées).
4. Modifier le programme `tsp.c` pour qu'il utilise une matrice des distances précalculée (à la place du calcul des distances euclidiennes à chaque évaluation d'une permutation). Construire la courbe C_3 sur le même principe que C_1 et C_2.
5. Donner vos conclusions sur les courbes C_1, C_2 et C_3.