1. Exercices 1

1.1 Il n'est pas possible de réaliser ce problème, si on compte le nombre d'intersection total pour chaque segment, on arrive à $5 * 3 = 15$, or, lors de chaque intersections, cela rajoute à chaque fois 2 intersections, donc on arrive jamais à 15 puisse que 15 est impair.

Avec 301 segments qui doivent en couper exactement 201 autres, cela nous donne $301 * 201 = 60501$, c'est impair dont impossible.

Pour résoudre ce problème on peut modéliser sous la forme d'un graphe : 5 sommets connectés à exactement 3 autres sommets.

1.2 Ajouter des nœuds supplémentaires avec un arbre de Steiner, l'arbre de Steiner ce construit à partir du diagramme de Voronoï.

1.4 Un problème est donnée par une matrice de flot F et une matrice de distance D. Si la matrice D est plus grande que F (si il y a plus de place que d'éléments à placer), alors on peut modifier la matrice F avec des éléments quelconque dont le coût est l'élément neutre.

1.5 $x_i \; in \; \{0, 1\}$ avec $\sum i \cdot x_i = 1170$ et $\prod i \cdot (1 - x_i) = 36000$ et donc on peut définir la fonction d'utilité $min(|\sum i \cdot x_i - 1170| - |\prod i \cdot (1 - x_i) - 36000|)$

1.6 On peut modéliser ce problème comme un problème de coloration de graphe.

1.9

2. Méthodes constructives

2.1 Construction aléatoire

Tirer aléatoirement une solution dans l'espace des solutions admissibles. L'avantage est que la méthode est très facile à implémenter mais la qualité de la solution est déplorable et un tirage aléatoire uniforme n'est pas évident à réaliser.

σ : permutation aléatoire $1..n$
σ_i : ième ville visité
$D = (d_{ij})$
minimiser $(\sum_{i=1}^{n-1} d_{\sigma_i \sigma_{i+1}}) + d_{\sigma_n \sigma_i}$

ou bien minimiser avec s_i est la ville qui suit la ville i

$$\sum_{i=1}^{n} d_{i s_i}$$

Data: Tableau de n element L
Result: Une permutation aléatoire de L
Définir l comme la longueur du tableau;
for *i allant de 1 à n* **do**
    Tirer aléatoirement $j \in [i; n]$;
    Permuter $L[j]$ avec $L[l]$;
    $l = l - 1$;
end

2.2 Méthode gloutonne

L'idée est de construire une solution élément par élément en ajoutant, à chaque pas, un élément approprié. Cela est optimal pour certain problème.

On part d'une solution s vide ou trivial. On a une fonction de coût incrémental qui mesure empiriquement l'adéquation d'ajouter l'élément e à s. Le fait d'ajouter un élément peut ajouter des contraintes sur les prochains éléments à ajouter.

; `/* Algorithme glouton en` $O(n^2)$ `*/`
Data: Une solution partielle minimal s // en général $\emptyset$
Data: $R = E$ // Ensemble des éléments pouvant être ajoutés à s
Result: Une solution gloutonne
while s *n'est pas une solution complète* **do**
  Calculer $c(e, s) \forall e \in R$;
  Choisir un e' optimisant $c(e, s)$;
  $s = s \vee e'$;
  ; `// propagation des contraintes`
  Supprimer de R tout les éléments qui ne peuvent être ajouter à s;
end

Il existe d'autre algorithme :

2.2.1 Regret maximum

Lors de chaque étape, choisir la ville e qui maximise la fonction

$$c(s, e) = min_{j,k \in R} d_{je} + d_{ek} - min_{j \in R} d_{ie} + d_{ej}$$

2.2.2 Meilleur insertion

Choisir la ville e qui minimise la fonction

$$c(s, e) = \text{coût d'insertion minimalde la ville e avec la tournée partiel s}$$

possible en maximisant : insertion de la ville la plus éloignée. Les deux méthodes sont en $O(n^2)$

3. Méthodes d'amélioration

Pseudo code d'une méthode d'amélioration locale :

; `/* Trame d'une méthode d'amélioration locale */`
Data: Une solution donnée (par exemple, à partir d'une construction gloutonne
Result: Une solution équivalente ou meilleure
do
  Essayer de trouver une amélioration;
  Faire l'amélioration trouver;
while *Une amélioration est effectué*;

Exemple de modification :
- Remplacer deux arêtes d'une tournée par deux autres
- **2-Opt** : inverser le sens de parcours d'une sous-chaîne (remplacer deux arêtes par deux autres)
- **3-Opt** : déplacer un chemin ailleurs dans la tournée (remplacer trois arcs par trois autres)
- **Or-Opt** : Déplacer une sous-chaîne de r sommet ailleurs dans la tournée avec $r = 3$ puis 2 puis 1 etc...

3.1 On arrive deux fois à -4 pour le premier chemin améliorant.

4. Méthodes aléatoires

4.1 Choix du prochain élément

Il existe plusieurs technique pour ce choix :

- GRASP : on calcule $c_{min} et c_{max}$ (coût d'insertion) et on choisis l'élément parmis un sous-ensemble R de E où E est l'ensemble des éléments disponibles et R est $\{r \in E | c_r \in [c_{min}; \alpha(c_{max} - c_{min})]\}$
- colonie de fourmie : le choix est inversement proportionnel au coût et les coût sont modifiés en fonction des solutions construites précédemment
- technique du bruitage : bruitage du coût en fonction d'une loi