Exercice

Nous voulons trouver des emplacements pour construire des casernes de pompiers dans une wilaya. Supposons qu'il y ait 5 villes différentes dans cette wilaya et 4 emplacements (locations) potentiels. Le temps de route (en minutes) entre chaque ville et les emplacements potentiels est donné par le tableau suivant. Sachant que le temps de route entre chaque ville et la future caserne qui la couvre ne doit pas dépasser 20 min, le problème est de trouver les meilleurs emplacements pour construire un nombre minimum de casernes tout en couvrant les 4 villes.

Ville / Emplacement	**1**	**2**	**3**	**4**	**5**
1	10	15	30	20	25
2	60	40	10	30	20
3	45	25	20	15	17
4	20	15	18	22	30

Questions :

1. Citer les grandes classes de résolution des problèmes d'optimisation, quels sont les avantages et les inconvénients de chaque classe ?
2. Écrire un modèle généralisé pour n emplacements et m villes qui trouve le nombre minimum de casernes à construire et leur emplacement, quelle est sa nature ?
3. Proposer un algorithme glouton pour résoudre ce problème, quel est le nombre de casernes trouvé ?
4. Expliquer comment résoudre ce problème exactement.
5. Proposer un algorithme de recherche locale pour résoudre ce problème en justifiant :
    a. La représentation d'une solution,
    b. La fonction d'évaluation utilisée,
    c. L'opérateur de modification de solution.
6. Quelles sont les limites des algorithmes de la recherche locale ? et comment les surmonter ?

Corrigé type

1. Citer les grandes classes de résolution des problèmes d'optimisation, quels sont les avantages et les inconvénients de chaque classe ?

On peut classer les méthodes d'optimisation en deux grandes classes :

1.1 **Classe des méthodes exactes :** cette classe contient les méthodes de résolution exactes, donc elles fournissent une (ou plusieurs) solution optimale.

- **Avantages**

  Preuve de l'exactitude de la solution optimale

0.5 pts

- **inconvénients**

- Problème d'explosion combinatoire (complexité spatiotemporelle énorme)
- Impraticable dans le cas des instances de grandes tailles.

1.2 **Classe des méthodes approchées :** contient les méthodes qui ne garantissent pas l'optimalité de la solution trouvée. Elle donne généralement une solution proche de l'optimale. Cette classe contient trois sous classes importantes : les algorithmes approximatifs, les heuristiques, et les métaheuristiques.

- **Avantage**

- Donne des solutions de bonnes qualités dans un temps raisonnable.
- Simple
- Fonctionnement intuitif

0.5 pts

- **Inconvénients**
  - Pas de garantie de l'optimalité
  - Faible ou manque de modélisation mathématique
  - Sensibilité au paramétrage

2. Écrire un modèle général pour n emplacements et m villes qui trouve le nombre minimum de casernes à construire et leur emplacement, quelle est sa nature ?
   - Définissant d'abord la variable de décision x_i qui prend la valuer 1 si l'emplacement i est utilisé pour construire une caserne, sinon 0 pour $i \in 1..n$.
   - Définissant également le paramètre $a_{ij} = 1$ si le temps entre la caserne i et la ville j est inférieur à 20, sinon 0 pour t=1 ..n et j=1..m.

Le modèle est le suivant :

$$\min nbrcasernes = \sum_{i}^{n} x_i \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (1)$$

0.50 pts

sc

$$\sum_{i}^{n} a_{ij} x_i \geq 1, \quad pour\ chaque\ j \in 1..m \dots\dots\dots\dots\dots\dots\dots (2)$$

0.25 pts

$$x_i \in \{0,1\} \dots i \in 1 \dots n \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3)$$

0.25 pts

- (1) est la fonction objectif qui consiste à minimiser le nombre de casernes
- (2) est la contrainte du problème qui assure que au moins une ville est assurée par au moins une caserne
- (3) la variable décision est binaire

0.25 pts

Quelle est sa nature ?

C'est un problème linéaire en nombres binaires

0.25 pts

3. Proposer un algorithme glouton pour résoudre ce problème, quel est le nombre de casernes à construire ?

L'idée de l'algorithme est la suivante :

1. Trier les emplacements disponibles selon le nombre de villes non couvertes les plus proches ; ainsi on obtient l'ordre suivant : 1 (3 villes),3 (3 villes),4 (3 villes),2 (2 villes)
2. Choisir le premier emplacement dans l'ordre ; emplacement 1 dans notre exemple, il couvre les villes 1, 2, et 4, ils nous restent les villes 3 et 5.
3. Recommencer à l'étape (1) avec les locations libres et les villes non encore couvertes ; emplacements 2,3,4 et les villes 3 et 5 jusqu'à la couverture de toutes les villes.

0.5 pts

L'application de cet algorithme nous donne comme solution les emplacements 1 et 3, donc le nombre de casernes à construire est 2.

0.25 pts

4. Expliquer comment résoudre ce problème exactement ?

L'algorithme à utiliser pour résoudre ce problème exactement est l'algorithme branch and bound. Les étapes de cet algorithme sont les suivantes :

1. Faire **la relaxation linéaire** de ce problème (P) et résoudre le problème relaxé par l'algorithme du simplexe 0.5 pts
2. Si la solution trouvée et binaire terminer l'algorithme et renvoyer la solution, sinon continuer (solution factionnée)
3. Stocker la valeur de la fonction trouvée par la relaxation dans la borne inferieure LB1, calculer la valeur de la borne supérieure UB par une heuristique ou la mettre ∞.
4. Choisir une variable fractionnée dans la solution du (P) et créer deux sous problèmes P1 et P2 en imposant : 0.25 pts
   $xi \leq round(x_i^*)$ ou $xi \geq round(x_i^*) + 1$ round: donne la valeur entière
   La solution optimale de P est la meilleure des solutions optimales des deux problèmes P1 et P2.
   $P1=\{ P, xi \leq round(x_i^*) \}$, $P2= \{ P, xi \geq round(x_i^*) +1\}$ 0.25 pts
5. Pour chaque sous problème
   a) Faire la relaxation linéaire de ce problème et résoudre le problème relaxé par l'algorithme du simplexe et calculer son LBj
   b) Si la solution trouvée et fractionnée **et** LBj >=UB alors
      Couper cette branche Fin si
   c) Si la solution trouvée et binaire **et** LBj >=UB alors
      Marquer ce nœud comme final (feuille) Fin si 0.25 pts
   d) Si la solution trouvée et binaire **et** LBj<UB alors
      MAJ de UB : UB=LBj Fin si
   e) Si la solution trouvée et fractionnée **et** LBj <UB alors
      Fractionner le problème courant en deux sous problème comme en étape 4 et recommencer à 5. Fin si

5. Si on veut résoudre ce problème d'une manière approchée par une méthode de recherche locale. Proposer un algorithme de recherche locale pour résoudre ce problème en spécifiant

   a) La représentation d'une solution

La représentation la plus facile est un vecteur binaire de taille n, où la valeur 1 d'une case i indique que l'emplacement est choisi, sinon il n'est pas choisi

1	0	1	1

Example d'une solution

0.5 pts

b) La fonction d'évaluation utilisée (expliquer bien)

Il faut d'abord vérifier la faisabilité de la solution, i-e elle couvre toutes les villes. Si elle n'est pas faisable, on lui donne un grand nombre (par exemple n+1) pour indiquer qu'elle s'agit d'une une mauvaise solution. Donc, La fonction d'évaluation d'une solution et :

$$\begin{cases} f(sol) = \sum_i^n sol(i) \text{ si la solution est réalisable} \\ \text{Sinon} \\ n+1 \text{ (ou } +\infty) \text{ pour l'écarter} \end{cases}$$

0.5 pts

c) Un opérateur de modification de solution

Un opérateur simple pour modifier une solution ***sol*** consiste à choisir aléatoirement quelques cases dans le vecteur solution puis de flipper leur valeur (0 devient 1, et 1 devient 0) comme il est montré par l'exemple suivant :

1	0	1	1
1	1	1	0

0.5 pts

L'algorithme de la recherche locale pour ce problème

Début

Lire nombre de location n

Lire nombre de villes m

Lire temps[n,m] % temps de route entre location i et ville j

Fixer itrmax % le nombre d'itérations de l'algorithme de recherche

Créer une solution initiale binaire **sol[1..n]**

%Calculer sa fonction d'évaluation

bestNb=f(sol) % la fonction doit vérifier que la solution est faisable

pour it=1 à itrmax faire

% créer une nouvelle solution à l'aide de l'opérateur de modification flip()

sol_{new}=flip(sol)

%calculer sa fonction d'évaluation

Scnew=f(sol_{new})

1 pts

% remplacer la solution en cours si la solution nouvelle est meilleure

Si Scnew < bestNb

Sol= sol$_{new}$

bestNb =Scnew

Fin si

Fin pour

Fin

5.Quelles sont les limites des algorithmes de la recherche locale ? et comment les surmonter ?

Les inconvénients majeurs de ces méthodes sont les suivantes :

- Grande probabilité de tomber un minimum local
- Lenteur
- Une exploration faible

0.25 pts, (il doit citer au moins deux)

Pour surmonter ces problèmes on peut soit :

- Ajouter des mécanismes d'échappement des minimas locaux comme dans le recuit simulé ou la recherche tabou
- Recommencer la recherche avec une autre solution un peu aléatoire

0.25 pts