

Étude de cas : k-couverture connexe minimum dans les réseaux de capteurs

BATY LÉO, BRUNOD-INDRIGO LUCA

3 novembre 2020

Table des matières

1	Calcul de bornes inférieures	2
1.1	Notations	2
1.2	Modèle PLNE	2
1.3	Relaxation linéaire et obtention des bornes inférieures	3
2	Heuristiques	4
2.1	heuristique 1	4
2.2	heuristique 2	5
3	Métaheuristiques	5
3.1	Structure de voisinage	5
3.2	Recuit simulé	5
3.3	Recherche à voisinages multiples	6

1 Calcul de bornes inférieures

Dans cette section, on pose le problème sous la forme d'un programme linéaire en nombres entiers (PLNE) et on en déduit des bornes inférieures par relaxation linéaire.

1.1 Notations

Voici tout d'abord les différentes notations qui seront utilisées dans la suite :

- $k \in \{1, 2, 3\}$
- $R^{capt} \leq R^{com}$
- T ensemble des cibles (targets), de cardinal n
 - $t \in T$, coordonnées (x_t, y_t)
 - $t, t' \in T, t \neq t', \Delta_{t,t'} = \sqrt{(x_t - x_{t'})^2 + (y_t - y_{t'})^2}$
- puit s de coordonnées (x_s, y_s)
- $E^{capt} = \{(t, t') \in T^2 \mid t \neq t', \Delta_{t,t'} \leq R^{capt}\}$
 - $E_t^{capt} = \{t' \in T \mid (t, t') \in E^{capt}\}$
 - **Graphe de captation** : $G^{capt} = (T, E^{capt})$
- $E^{com} = \{(t, t') \in (T \cup \{s\})^2 \mid t \neq t', \Delta_{t,t'} \leq R^{com}\}$
 - $E_t^{com} = \{t' \in T \mid (t, t') \in E^{com}\}$
 - **Graphe de communication** : $G^{com} = (T \cup \{s\}, E^{com})$
 - On note $\mathcal{D}(E^{com})$ l'ensemble des arrêtes orientées contenant les (t, t') et (t', t) .
- $\forall t \in T, \delta_t = \mathbb{1}_{\{\text{capteur sur la cible } t\}}$
- $\forall e \in E^{com}, x_e = \mathbb{1}_{\{e \text{ dans l'arbre de communication}\}}$

1.2 Modèle PLNE

On décrit une solution à l'aide des variables de décision suivantes :

- $\forall t \in T, \delta_t = \mathbb{1}_{\{\text{capteur sur la cible } t\}}$
- $\forall e \in E^{com}, x_e = \mathbb{1}_{\{e \text{ dans l'arbre de communication}\}}$ (l'arbre de communication est un arbre couvrant toutes les cibles possédant un capteur dans le graphe de communication G^{capt})
- $\forall a \in \mathcal{D}(E^{com}), y_a \in \mathbb{R}_+$: on définit un flot sur $\mathcal{D}(E^{com})$, qui part de s , de valeur égale au nombre de capteurs placés, et dont chaque cibles possédant un capteur réceptionne une unité.

Afin que les contraintes définies ci-après soient valables aussi pour la source s on fixe $\delta_s = 1$:

- On minimise le nombre de capteurs placés : (1a)
- k-connexité : (1b)
- nombre d'arêtes de l'arbre fixé à n : (1c)
- Une arête n'appartient pas à l'arbre couvrant si au moins une de ses deux extrémités ne possède pas de capteur : (1d), et (1e)

- Pour chaque cible, la différence entre le flot entrant et le flot sortant vaut 1 si elle possède un capteur, et 0 sinon : (1f)
- Le différence entre le flot sortant et le flot entrant dans la source est égal au nombre de cibles placées : (1g)
- On interdit le flot sur les arcs dont l'arête associées n'est pas dans l'arbre de communication entre les capteurs : (1h) et (1i)

Voici le PLNE complet obtenu :

$$\min_{\delta, x, f} \sum_{t \in T} \delta_t \quad (1a)$$

$$\text{s.t.} \quad \sum_{t' \in E_t^{capt}} \delta_{t'} \geq k, \quad \forall t \in T \quad (1b)$$

$$\sum_{e \in E^{com}} x_e = n, \quad (1c)$$

$$x_e \leq \delta_t, \quad \forall e = (t, t') \in E^{com} \quad (1d)$$

$$x_e \leq \delta_{t'}, \quad \forall e = (t, t') \in E^{com} \quad (1e)$$

$$\sum_{t' \in E_t^{com}} y_{(t', t)} - \sum_{t' \in E_t^{com}} y_{(t, t')} = \delta_t, \quad \forall t \in T \quad (1f)$$

$$\sum_{t \in E_s^{com}} y_{(s, t)} - \sum_{t \in E_s^{com}} y_{(t, s)} = \sum_{t \in T} \delta_t, \quad (1g)$$

$$y_{(t, t')} \leq n \times x_e, \quad \forall e = (t, t') \in E^{com} \quad (1h)$$

$$y_{(t', t)} \leq n \times x_e, \quad \forall e = (t, t') \in E^{com} \quad (1i)$$

$$\delta_t \in \{0, 1\}, \quad \forall t \in T \quad (1j)$$

$$x_e \in \{0, 1\}, \quad \forall e \in E^{com} \quad (1k)$$

$$y_a \geq 0, \quad \forall a \in \mathcal{D}(E^{com}) \quad (1l)$$

1.3 Relaxation linéaire et obtention des bornes inférieures

Nous avons implémenté le PLNE en JULIA en utilisant la librairie JUMP (librairie de programmation mathématique) couplée avec le solveur Gurobi (licence académique), afin de résoudre la relaxation linéaire de ce problème, voire la résolution exacte par Branch and Bound pour les plus petites instance. Nous obtenons exactement les mêmes bornes inférieures que celles fournies avec les instances, sauf pour les plus grosses instances pour lesquelles le temps de calcul était trop important pour pouvoir les calculer sur nos ordinateurs.

2 Heuristiques

2.1 heuristique 1

Algorithm 1 Heuristique par captations de coût minimal successives

Input :

$(T, G^{capt}, G^{com}, k)$: une instance du problème

L : une liste contenant k fois chaque cible de T dans un ordre quelconque

Output : $N \subset T$: le sous-ensemble de cibles sur lesquelles placer des capteurs

$N = \emptyset$

for all $t \in L$ **do**

 Choisir t' tel que $(t, t') \in E^{capt}$ et qui minimise le plus court chemin vers $s \cup N$ dans G^{com}

$E^{capt} = E^{capt} \setminus (t, t')$

 Ajouter à N les cibles situées sur le plus court chemin de t' à $s \cup N$ dans G^{com}

end for

Renvoyer N

2.2 heuristique 2

Algorithm 2 Heuristique gloutonne avec relation d'ordre sur les cibles

Input :

$(T, G^{capt}, G^{com}, k)$: une instance du problème

\prec : une relation d'ordre sur les cibles

Output : $N \subset T$: le sous-ensemble de cibles sur lesquelles placer des capteurs

$N = \emptyset$

$Q = \{t \in T \mid (s, t) \in E^{com}\}$

$U = \emptyset$

while Toutes les cibles n'ont pas au moins k voisins appartenant à N dans G^{capt} **do**

if $Q \setminus U \neq \emptyset$ **then**

 Choisir t dans $Q \setminus U$ maximal pour la relation d'ordre \prec

if t a au moins un voisin dans G^{capt} ayant moins de k voisins dans N **then**

$Q = Q \setminus \{t\}$

$Q = Q \cup \{t' \mid (t, t') \in E^{com}\}$

$N = N \cup \{t\}$

else

$U = U \cup \{t\}$

end if

else

 Choisir t dans $Q \cap U$ maximal pour la relation d'ordre \prec

$Q = Q \setminus \{t\}$

$Q = Q \cup \{t' \mid (t, t') \in E^{com}\}$

$N = N \cup \{t\}$

end if

end while

Renvoyer N

3 Métaheuristiques

3.1 Structure de voisinage

3.2 Recuit simulé

Afin d'améliorer les solutions obtenues à l'aide de l'heuristique 1, nous avons mis en œuvre une métaheuristique de type recuit simulé comme vu en cours, en utilisant le voisinage défini par la seconde heuristique 2. Nous avons utilisé les paramètres suivant :

- Paramètre multiplicateur de **décroissance de la température** : $\phi = 0.95$
- **Nombre d'itérations** par palier de température : `number_iterations` = 10000

- **Température initiale** : nous avons fixé un paramètre correspond à la probabilité initiale de garder une modification qui dégrade l'objectif à $\text{initial_keep_probability} = 0.8$. Puis en calculant une moyenne empirique de la valeur de dégradation d'une solution, on en déduit une

valeur de la température initiale :
$$T_0 = \frac{-\Delta_{mean}}{\ln(\text{initial_keep_probability})}$$

- **Température finale** : de manière similaire au calcul la température initiale on a choisi $\text{final_keep_probability} = 1e-10$ qui correspond à la probabilité finale d'accepter une solution

qui augmente l'objectif de 1. On obtient donc
$$T_{final} = \frac{-1}{\ln(\text{final_keep_probability})}$$

En sortie du recuit, on applique en plus une recherche locale. Voici les résultats finaux obtenus, et leur comparaison avec les bornes inférieures :

TABLE 1 – Comparaison entre : résultats du recuit simulé | bornes inférieures

(k, R ^{capt} , R ^{com})	150-7-4	225-8-10	625-12-100	900-15-20	1500-15-100	1500-18-100
(1, 1, 1)	31 21	45 26	89 54	137 82	138 80	198 113
(1, 1, 2)	19 19	25 25	55 53	83 80	88 80	121 112
(1, 2, 2)	8 6	11 8	24 16	37 23	37 23	55 32
(1, 2, 3)	6 6	8 8	18 16	27 22	28 23	39 31
(2, 1, 1)	44 40	56 50	119 106	185 162	189 159	261 225
(2, 1, 2)	39 39	50 50	109 106	166 161	174 159	239 225
(2, 2, 2)	12 11	16 15	34 31	52 44	53 45	73 62
(2, 2, 3)	11 11	16 15	32 31	49 44	52 45	69 62
(3, 1, 1)	62 60	79 78	166 162	253 246	260 239	355 341
(3, 1, 2)	60 60	78 78	166 162	251 245	257 239	355 341
(3, 2, 2)	17 17	23 23	49 46	74 66	75 67	103 93
(3, 2, 3)	17 17	23 23	48 46	72 66	75 67	105 93

Remarque : pour l'instance 1500-18-100, on obtient une solution à 103 pour le jeu de paramètres (3, 2, 2) et 105 pour le jeu de paramètres (3, 2, 3). Toute solution de (3, 2, 2) étant admissible pour (3, 2, 3), on a donc aussi une solution à 103 pour (3, 2, 3).

3.3 Recherche à voisinages multiples

Afin d'affiner et d'améliorer de manière très légère les résultats du recuit simulé, nous avons implémenté une recherche à voisinages multiples, qui part des solutions du recuit, et qui alterne entre des 2-opt, 3-opt, et 4-opt, cela pendant 500000 itérations. On a observé une amélioration des solutions qui varie entre 0 et 5. Voici une table résumant les résultats obtenus, avec en couleur les jeux de paramètres/instances pour lesquels on a amélioré la solution d'au moins 1.

expliquer l'algo ?

TABLE 2 – Comparaison entre : résultats finaux | bornes inférieures

$(\mathbf{k}, \mathbf{R}^{\text{capt}}, \mathbf{R}^{\text{com}})$	150-7-4	225-8-10	625-12-100	900-15-20	1500-15-100	1500-18-100
(1, 1, 1)	31 21	45 26	88 54	137 82	137 80	196 113
(1, 1, 2)	19 19	25 25	55 53	82 80	87 80	121 112
(1, 2, 2)	8 6	11 8	24 16	36 23	36 23	53 32
(1, 2, 3)	6 6	8 8	18 16	26 22	27 23	38 31
(2, 1, 1)	44 40	56 50	117 106	183 162	184 159	259 225
(2, 1, 2)	39 39	50 50	109 106	165 161	170 159	237 225
(2, 2, 2)	12 11	16 15	34 31	50 44	53 45	71 62
(2, 2, 3)	11 11	15 15	32 31	48 44	50 45	69 62
(3, 1, 1)	61 60	79 78	165 162	253 246	255 239	354 341
(3, 1, 2)	60 60	78 78	166 162	249 245	253 239	352 341
(3, 2, 2)	17 17	23 23	49 46	73 66	75 67	102 93
(3, 2, 3)	17 17	23 23	48 46	72 66	74 67	102 93