le cnam

MPRO

Projet de Métaheuristique

MH

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

---

Antoine Desjardins
Martin Lainée

Table des matières

1 Introduction **2**

2 Minorant **2**

3 Modélisation **2**
3.1 Cas des Grilles 2
3.2 Cas du placement Aléatoire 2

4 Solutions admissibles **2**
4.1 Arbre couvrant 2
4.2 Solution gloutonne (meilleure) 3

5 Algorithme Génétique **3**
5.1 Réparation 4
5.2 Optimisation 4
5.3 Mutation 4
5.4 Basique colonnes 4
5.5 Basique carré 4
5.6 Mixte 6
5.7 Ouverte carré 6

6 Expériences numériques **6**

1 Introduction

On peut effectuer quelques remarques sur le problème de combinatoire posé.

— On ne s'intéresse qu'aux cas où $Rcom > R_{capt}$.

— Le problème n'admet pas nécéssairement de solution. En effet on ne peut pas place plus d'un capteur par cible, donc on peut majorer k par $R_{capt}^2 + R_{capt}$ (cible située dans un coin de la grille)

— si les rayons sont inférieurs à 2, alors la norme 1 et la norme 2 sont équivalentes dans le cas d'une topologie de grille

— Dans le cas de 1-connexité de rayons $(1, 1)$, Le problème se résume à un problème d'arbre couvrant de poids minimum de racine le puit et dont on peut supprimer les feuilles de degré 1. Si $k > 1$, alors on peut au mieux remplir les feuilles de degré 1. Si la solution n'est pas admissible, alors le problème n'admet pas de solution.

— Dans le cas plus général, on retrouve un genre de problème d'arbre couvrant dans une topologie inusuelle.

2 Minorant

On serait tenté de calculer un minorant à l'aide d'une représentation géométrique simple, en divisant simplement $k*$ la surface à capter par la surface captée par un capteur. Ceci fonctionne pour une grille (en enlevant les "trous"), mais pas pour un placement aléatoire, car dessiner simplement l'enveloppe convexe des pôints peut mener à une surestimation du coût (on demande de capter des zones blanches). Nous nous contenterons donc des bornes fournies pour évaluer la qualité de nos résultats.

3 Modélisation

3.1 Cas des Grilles

On modélise le graphe par une matrice carrée qui contient les valeurs :

— -2 si le point n'appartient pas au graphe

— -1 si le point n'est pas capté et n'est pas à distance de communication d'un capteur

— 0 si le point n'est pas capté et est à distance de communication d'un capteur

— k si le point est capté par k capteurs, et est donc à distance de communication d'un capteur

On stocke de plus sur chaque point si un capteur y est posé ou non.

3.2 Cas du placement Aléatoire

On modélise le problème à l'aide d'une représentation de l'adjacence sous deux formes différentes.

D'une part, on construit les matrices d'adjacence *Adjcom* et *Adjcapt* qui représente les cibles qui sont à distance de communication (respectivement de communication) l'une de l'autre.

D'autre part, on construit une liste des emplacement avec, pour chaque numéro de capteur :

— -1 ou 1 selon qu'un capteur est placé sur l'emplacement ou non

— pour chacune des distances 1,2 et 3, la liste des cibles placées à cette distance (ou moins) de l'emplacement considéré. Ce stockage est assez volumineux mais nous épargnera des lectures de matrice d'adjacence redondantes plus tard.

— le nombre de fois où la cible est captée dans la solution actuelle

— ses coordonnées (x,y)

4 Solutions admissibles

4.1 Arbre couvrant

Afin de trouver un majorant du problème, on va considérer un arbre couvrant sur le graphe des adjacences par distance de captation.

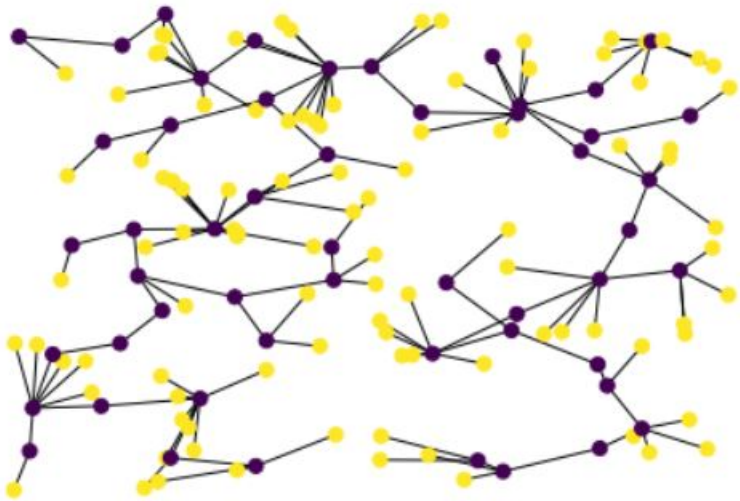

FIGURE 1 – Arbre couvrant de poids minimum et ses feuilles

Nous allons tracer en $O(Elog(V))$, à l'aide de l'algorithme de Kruskal, un arbre couvrant de poids minimum sur le graphe compasé des emplacements (sommets) et des arêtes des sommets qui sont à R_{com} l'un de l'autre. Nous allons ensuite supprimer les feuilles de l'arbre (de profondeur 1) afin d'avoir un arbre couvrant de poids minimum qui peut capter tous les emplacements. Bien entendu, tout algorithme permettant de calculer un arbre couvrant de poids minimal convient (par exemple, l'algorithme de Prim). Dans la mesure où $Rcom > R_{capt}$, tous les capteurs communiquent bien entre eux. Cette solution ne marche que pour $K = 1$.

4.2 Solution gloutonne (meilleure)

Pour obtenir une solution admissible, on procède de manière gloutonne en choisissant une d'abord une cible aléatoirement et uniformément dans le graphe, sur laquelle on place un capteur. On place ensuite les capteurs un à un sur le réseau tels que :

— Le nouveau capteur est à distance de captation d'au moins un autre capteur

— Le nombre de cibles pas encore couvertes par k capteurs dans la zone de captation du nouveau capteur est maximal, choisissant de manière indifférente en cas d'égalité.

Une fois toutes les cibles couvertes par k capteurs, si aucun capteur n'est à distance R_{com} du puit, on rajoute un capteur pour rendre l'intégralité du réseau connexe pour la communication. Une méthode alternative de construction du réseau est de maximiser à chaque ajout le nombre de cibles couvertes par le moins de capteur, pour couvrir en priorité les cibles peu couvertes.
Au terme de la construction, on peut effectuer une descente de gradient selon la fonction de voisinage qui consiste à chercher les cibles dont la connexité est plus élevée que nécéssaire et chercher si, dans les capteurs qui la capte, il en existe qui peuvent être retirés sans porter atteinte à l'admissibilité de la solution.

5 Algorithme Génétique

Au vu de la structure du problème, nous avons voulu essayer d'y appliquer une méta-heuristique de population, en l'occurence un algorithme génétique.

— L'ADN constitue la solution en elle-même, i.e. la liste des cibles où l'on place un capteur

— Les chromosomes représentent des régions de l'espace (et les cibles, occupées par un capteur ou non, qui s'y trouvent)

— Lorsqu'un enfant hérite de deux parents, on dit que chaque parent lui donne son chromosome associé à une même région de l'espace. Pour l'enfant, ces deux chromosomes deviennent deux allèles dont l'une seulement sera dominante et exprimée ; l'enfant choisit, pour cette région de l'espace (ce chromosome) la solution d'un parent seulement, de façon aléatoire.

— Nous appelons première génération la génération spontanée née de nos solutions admissibles gloutonnes.
— La reproduction a lieu entre tous les couples de N parents, formant N^2 enfant. Ainsi les parents se retrouvent également dans la génération suivante (identique à un processus de clonage ou reproduction asexuée).
— La mutation a lieu losque que l'ADN de l'enfant est muté par une fonction de voisinage
— La réparation correspond à la réparation de solutions non admissibles par ajout de capteurs
— L'optimisation correspond à l'optimisation locale d'un individu en supprimant ses cibles redondantes sans casser son admissibilité (viabilité)
— La sélection naturelle s'effectue en éliminant les enfants non viables (solutions non admissibles), puis les enfants aux solutions les plus mauvaises (on ne garde que les N plus forts).

Notre processus s'effectue comme suit :

— Génération spontanée
— Reproduction (croisement et reproduction asexuée)
— éventuelle mutation
— réparation des enfants ne captant pas toutes les cibles K fois en ajoutant des cibles de façon gloutonne
— sélection des enfants viables (élimination des enfants non viables, i.e. non connexes en communication)
— Optimisation
— sélection naturelle des N plus forts
— reproduction, etc

5.1 Réparation

Afin de réparer une solution non admissible qui ne capte pas k fois toutes les cibles, nous ajoutons de façon gloutonne (comme dans la génération de solutions) des capteurs afin de capter les cibles de vant l'être jusqu'à ce que la solution soit admissible.

Certaines solutions ne sont pas admissibles car leurs capteurs ne sont pas connectés au puit. Nous supprimons ces solutions car les réparer serait a priori coûteux en termes de calculs.

5.2 Optimisation

On optimise les solutions avec un genre de descente de gradient local, i.e. on supprime tous les capteurs qui peuvent l'être. Cela permet d'aller voir au fond de chaque vallée représentée par une solution située dedans.

5.3 Mutation

Nous n'avons pas implémenté de mutations, mais voici à titre d'exemples plusieurs possibilités que l'on pourrait tester :

— Ajouter k capteurs, puis supprimer d'autres capteurs en conservant la validité du réseau
— Supprimer k capteurs, puis ajouter des capteurs différents jusqu'à obtenir un réseau valide.
— Déplacer un capteur

5.4 Basique colonnes

Algorithme génétique sans mutations. L'espace est découpé en [nombre de chromosomes] colonnes égales. Le croisement entre parents s'effectue de façon indépendante sur chaque colonne.

5.5 Basique carré

Algorithme génétique sans mutations. L'espace est découpé en [nombre de chromosomes] carrés égaux (ce nombre doit être un carré parfait). Le croisement entre parents s'effectue de façon indépendante sur chaque colonne.

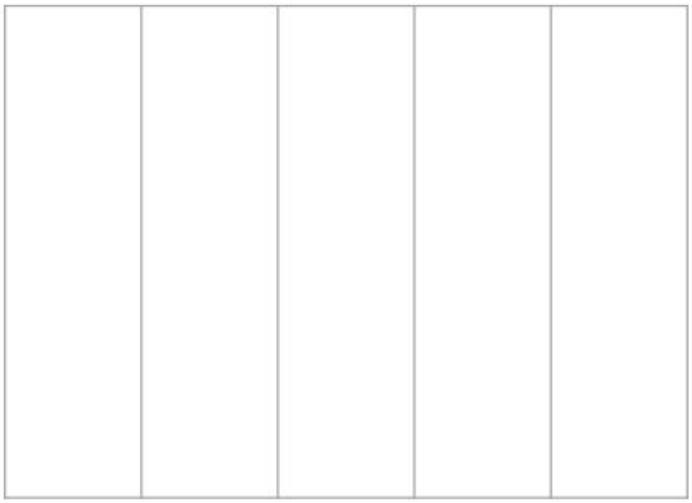

FIGURE 2 – Répartition de l'espace en colonnes égales

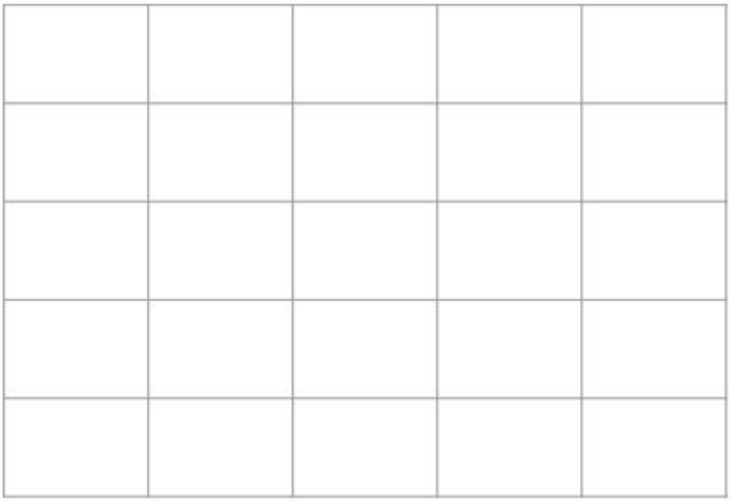

FIGURE 3 – Répartition de l'espace en carrés égaux

5.6 Mixte

Algorithme génétique sans mutation. Le croisement s'effectue en alternant entre les méthodes précédentes.

5.7 Ouverte carré

Le principe est le même que pour la méthode basique, mais en introduisant un certain nombre d'individus générés aléatoirement à chaque génération.

6 Expériences numériques

Dans cette section nous expliquons ce qui ressort de nos expériences et ce qui nous a mené à sélectionner notre métaheursitique et nos paramètres pour l'évaluation.

Commençons par expliciter le rôle des hyper-paramètres :

— N est le nombre constant d'individus à chaque génération juste avant la reproduction

— T représente le nombre de générations avant arrêt du programme

— $nchr$ représente le nombre de chromosomes (i.e. le nombre de zones de l'espace considérées)

Augmenter N est toujours strictement meilleur, mais le coût de cette augmentation est quadratique. En restant aux alentours de 15, on obtient un temps d'exécution de chaque génération qui reste raisonnable, sans compromettre la qualité de la solution.

En raison du caractère aléatoire de la réparation des solutions, augmenter le nombre T de générations peut théoriquement continuer à améliorer la solution à l'infini. On choisit donc ce paramètre en fonction du temps d'exécution souhaité

Le nombre de chromosome n'augmente pas de manière significative le temps de calcul. Cependant, au-delà d'un certain stade, il est inutile de l'augmenter. Ceci est du au fait que de trop nombreux chromosomes suppriment la cohérence locale du génotype que nous avons adopté. L'idéal est de garder des zones géographiques de l'ordre de D_{capt} la distance de captation, afin de permettre à la réparation des solutions d'atteindre l'intérieur de chaque zone.

Le choix des chromosomes carrés plutôt que d'autres est justifié par le fait qu'à nombre de chromosomes similaires, les performances sont équivalentes. Cependant, ce choix de chromosome permet à toute les zones géographiques représentées par le chromosome d'être de l'ordre de grandeur de D_{capt}.

Le choix de l'heuristique mixte semble légèrement meilleur, les résultats étant en général meilleurs pour un même nombre de générations.

L'heuristique ouverte permet une meilleure diversification des solutions, au prix de la qualité des solutions que l'on évalue. Sur les expériences que nous avons réalisées, la différence ne semble pas très visible, et nous n'avons pas la puissance de calcul pour faire fonctionner avec un nombre d'individus significativement plus grand.