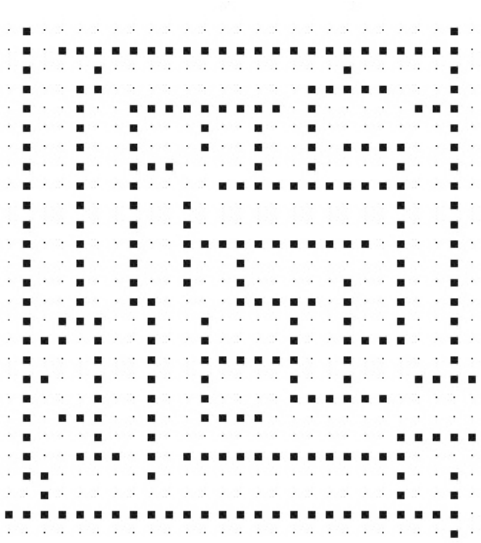
MPRO

Mathieu MARI - Augustin PAUCHON

Couverture connexe minimum d'un réseau de capteurs

Professeur : Agnès PLATEAU

Cedric - CNAM



Résumé.

Ce rapport présente la tentative de résolution d'un problème de couverture connexe minimale par une méta-heuristique génétique. Nons montrerons les arguments qui nous ont poussé à choisir un telle heuristique, nous présenterons les résultats obtenues et discuterons du choix des différents paramètres qui interviennent dans l'algorithme.

La créativité et le génie ne peuvent s'épanouir que dans un milieu qui respecte l'individu et célèbre la diversité.
Tom Alexander.

10 Octobre 2016

1 Présentation du problème

Le problème étudié ici consiste à trouver une couverture connexe minimum d'une grille de taille $N * N$ d'un réseau de capteurs, chauque capteur étant placé sur un point de la grille. Le nombre de capteur doit donc être minimisé en vérifiant les contraintes suivantes :

(couverture) chaque point de la grille (à l'exception du point $(0, 0)$) est dans le rayon de captation (Rc)[1] d'au moins un capteur.

(connexité) chaque capteur peut communiquer avec le puit $(0, 0)$ via un chemin de capteurs dans lequel deux capteurs sont adjacent s'ils sont distant d'au plus Rb.

2 Méta-heuristique évolutionnaire

2.1 Principe général

On veut mettre en place un algorithme évolutionnaire pour résoudre ce problème. On veut donc construire une suite de générations de solutions $(G_i)_{i\in\mathbb{N}}$. Chaque génération G_i sera composée de S solutions admissibles (*individu*). Pour passer d'une génération à la suivante, on utilise plusieurs principes :

1. La *reproduction* entre individus.
2. Des *mutations.*
3. La *séléction* des individus présentant les meilleures qualités (ici un nombre de capteur minimal).

L'algorithme s'arrête selon un *critère d'arrêt.* On renvoi alors le meilleur[2] individu rencontré au cours de l'algorithme.

2.2 Motivations

Plusieurs observations nous ont conduit à penser qu'une méta-heuristique évolutionnaire serait adaptée à la résolution de ce problème.

- Une méta-heuristique basée sur des déplacement de capteurs ne semble pas adaptée (et donc aussi l'exploration de *voisinages*), car déplacer un capteur ne diminue pas *a priori* le nombre de capteurs de la grille, et on risque d'engendrer des solutions non admissibles.
- Il est difficile de mettre en place une heuristique de réparation efficace. Dans notre méta-heuristique, c'est très rare[3] qu'une grille soit non admissible et dans ce cas elle est supprimée à la génération suivante, ce qui en outre n'a pas de conséquences notable sur l'efficacité de l'algorithme.
- Les minimums locaux risquent d'être relativement "éloignés" et il semble intéressant de parcourir l'espace des solutions à la fois "en profondeur" (trouver de bonnes grilles) et "en largeur" (encourager la diversité).
- Il faut faire en sorte que le nombre de capteurs des grilles tende à être modifié, de préférence diminuer (ce sera la sélection) mais aussi augmenter.

2.3 Génération d'une population initiale

Afin de produire la première génération, on utilise une heuristique gloutonne.

On génère des solutions de manière gloutonne en partant d'une grille pleine de capteurs puis en supprimant des capteurs de manière aléatoire jusqu'à ce qu'il ne soit plus possible d'en supprimer un sans violer une des deux contraintes.

remarque : Cette heuristique est relativement performante lorsque les valeurs du couple (Rc, Rb) sont élevées. Il arrive de trouver une solution optimale parmi les individu de la première génération.

[1] la distance utilisée est la norme Euclidienne.
[2] la grille qui possède le moins de capteur.
[3] d'un point de vue des probabilités

2.4 Détails des notions

La reproduction. L'algorithme fait intervenir la notion de reproduction. A partir de deux grilles *père* et *mère*, on va générer deux nouvelles grilles, *fils 1* et *fils 2*. Pour ce faire, on découpe verticalement la grille en deux selon un axe choisi aléatoirement. Le fils 1 (resp. fils 2) est construit en copiant les capteurs du père à gauche (resp. droite) de cet axe et ceux de la mère présents à droite (resp. gauche).

Il se peut que la grille obtenue ne soit pas admissible. Pour palier ce problème on définit une bande de superposition où les capteurs des deux parents sont présents. Si la largeur de la bande est supérieure à R_b, le fils est nécessairement admissible.

Cette bande risque d'être surchargée de capteur, c'est pourquoi nous appliquons à chaque fils une heuristique (gloutonne) de simplification, éliminant les capteurs inutiles. Il n'est donc pas gênant de choisir une bande épaisse, cela revient à inclure le principe de mutation dans celui de reproduction!

Le nombre de reproductions effectuées à chaque génération n'influence pas le déroulement de l'algorithme. Nous l'avons choisi égal à S.

La sélection. Lorsque qu'une reproduction est effectuée entre deux individus est effectué, on calcule le nombre de capteur des deux fils obtenus et on les compare à ceux des parents. Pour la génération suivante, on en garde alors deux parmi les quatre en suivant les règles suivantes.

- Si les enfants sont meilleurs que leurs parents, on remplace les parents par leur progénitures.
- Si les deux parents sont meilleurs, on leur fait éventuellement[4] une mutation.

Dans une première version de l'algorithme, nous avions imaginé un autre principe de sélection. Après un certain nombre de reproductions, on sélectionnait les S meilleurs individus parmi l'ensemble des parents et de leurs enfants. Mais ceci n'était finalement pas très adapté car il arrivait que les deux parents et leurs deux enfants soient sélectionnés dans la génération suivante. Or, la reproduction entre un parent et un de ses descendants est à proscrire car les individus obtenus suite à un inceste ressembleront beaucoup à leurs parents incestueux. ce type de sélection empêchait la diversité et l'algorithme focalisait alors sa recherche sur un parcours en profondeur : Un individu particulier performant (peu de capteurs) transmettait ses gènes progressivement à l'ensemble de la population qui finissait tous par lui ressembler.

Il est donc important pour la diversité de faire une sélection après chaque reproduction.

Les mutations. Comme on vient de le voir, les mutations sont déjà observables lors des reproductions, lorsque l'on choisit une large bande de superposition.

Le deuxième type de mutation consiste à rajouter aléatoirement un certain nombre de capteurs à une grille donnée, puis d'appliquer l'heuristique de simplification (gloutonne). Pour savoir à quel moment appliquer cette mutation, on associe à chaque grille une durée de vie[5], initialement nulle lorsque la grille est créée/issue d'une mutation. Lorsque suite à une reproduction, un parent est sélectionné dans la génération suivante (parce que ses fils sont moins bons que lui), sa durée de vie augmente d'une unité. Lorsque qu'un individu a vécu suffisamment longtemps (sa durée de vie a dépassé la durée de vie maximale autorisée), on peut considérer qu'il a suffisamment contribué à répandre ses gènes parmi les individus et il subit alors une mutation. Le choix d'une durée de vie maximale faible favorise la diversité, mais il faut faire attention à ce qu'il ne soit pas trop faible, sinon le parcours en profondeur devient compromis (ce serait dommage d'éliminer si vite une bonne grille!). Dans la pratique, on s'est rendu compte qu'une durée de vie maximale de 5 est un bon compromis.

Critère d'arrêt. Comme nous n'avons pas d'informations *a priori* sur la valeur optimale du problème, nous avons basé notre critère d'arrêt sur la stagnation du la solution optimale. Si la meilleure grille rencontrée jusqu'alors est inchangée depuis plus de τ générations, l'algorithme s'arrête et on renvoie cet individu. Choisir un τ élevé nous donne davantage de garanties sur l'optimalité de la solution renvoyée mais allonge également le temps de calcul.

3 Implémentation en C++

3.1 La classe `Case`

Cette classe permet de représenter le contenu de chaque position de la grille. Attributs de la classe :

[4] voir le paragraphe suivant
[5] un entier

- `capteur` : booléen indiquant la présence d'un capteur dans la case considérée.
- `degreCase` : entier donnant le nombre de capteur de la grille présents dans un rayon inférieur à R_c.
- `voisins` : liste des positions (représentée par une paire d'entier), des capteurs capables de communiquer avec la case considérée.
- `marque` : booléen permettant de marquer la case (utilisé pour le parcours du graphe des connections).

3.2 La classe `Grille`

Les objet de la classe grille représente les grilles que l'on veut couvrir de capteur. Attributs de la classe :

- `cases` : tableau de d'objet de la classe case représentant la grille.
- `listeCapteurs` : liste des positions des cases contenant un capteurs
- `jauge` : entier évaluant le coût d'une grille donnée, dans le cas d'une grille connexe il s'agit donc du nombre de capteur de la grille, sinon la jauge vaut N^2.
- `vie` : entier donnant le nombre d'utilisation de la grille pour générer une nouvelle grille (permet d'éviter que ses gènes soit omniprésents dans les différentes générations en effaçant la grille lorsque sa vie dépasse un certain seuil).

La classe Grille est munie de différentes méthode permettant d'ajouter, de supprimer ou d'ajouter des capteur à une position donnée. Une méthode permettant d'évaluer la connexité à aussi été implémentée.

Connexité d'une grille L'évaluation de la connexité se fait en parcourant une fois le graphe des capteurs : En partant du puit, on parcours sur chaque sommet la liste de ses voisins (donné par l'attribut `voisins`), et ce sommet, s'il n'est pas marqué est mis en "attente" dans une pile. Le sommet dont tout les voisins ont été parcouru est alors marqué puis on itère le processus sur le dernier élément empilé. Lorsque la pile est vide, cela signifie que la composante connexe contenant le puit à été totalement parcourue. Il suffit alors de vérifier que le nombre de capteurs marqués correspond au nombre de capteurs total pour en déduire la connexité du graphe des capteurs.

3.3 La classe Generation

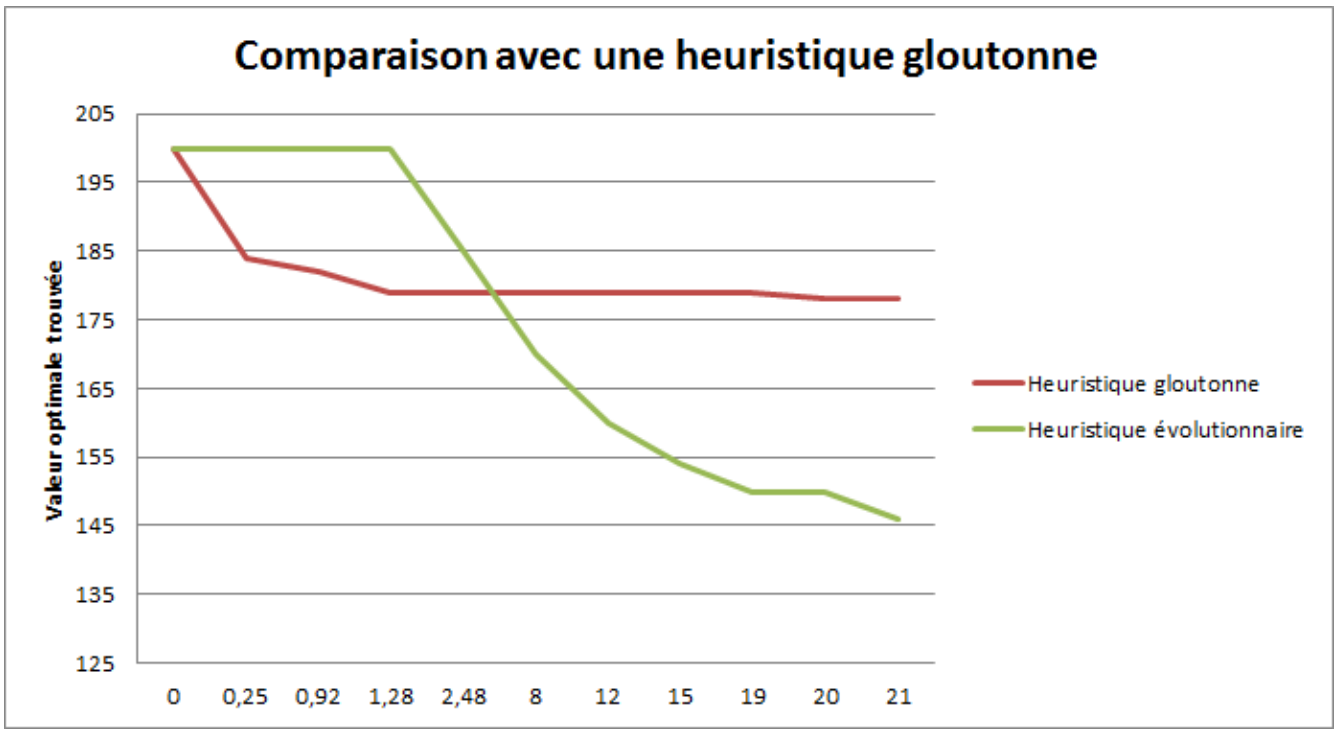
Les objet de cette classe représentent une tribu de S grilles; Attributs de la classe :

- `tribu` : tableau de taille S contenant des objets de type Grille.
- `chef` : objet de type Grille, meilleur individu de la tribu.
- `vmin` : valeur optimal obtenu
- `annee` : enregistre le nombre de renouvellement des générations.
- `stagnee` : donne le nombre de génération consécutive sans amélioration de la valeur optimal.

4 Résultats et tests d'efficacité.

4.1 Comparaison avec l'heuristique gloutonne.

Pour vérifier que l'algorithme génétique est plus efficace que l'heuristique consistant à générer des grilles de façon gloutonne et d'en regarder la meilleur, des tests on été réalisé et le graphique suivant présente les résultats.



4.2 Bornes inférieures théoriques

Dans cette partie, on donne quelques bornes inférieures du nombres de capteurs minimal nécessaires, en fonction de N, R_c et R_b. Il n'est malheureusement pas possible (sauf éventuellement si P=NP) de trouver une "formule miracle" décrivant ces optimaux, mais on peut quand même essayer d'en trouver de "pas trop mauvaises".

Cas $(R_c, R_b) = (1, 1)$ et $3/N$. Lorsque N est un multiple de 3, on sait trouver la borne inférieure optimale :

$$N + \frac{N^2 - N}{3}$$

Il suffit de concevoir un arbre ne possédant que $\frac{N}{3}$ intersections.

Cas général. On note B_R l'ensemble des points entiers du plan contenus dans la boule unité.

$$B_R = \{p, q \in \mathbb{Z}^2, p^2 + q^2 \leq R^2\}$$

et on note K_R son cardinal.

R	1	2	3	4
K_R	5	13	29	59

Soit b le nombre de capteurs d'une couverture connexe d'une grille de taille N avec les paramètres R_c et R_b. La contrainte de couverture se traduit par :

$$b \cdot K_{R_c} \geq N^2$$

La contrainte de connexité se traduit par le fait que pour chaque capteur, il existe un autre capteur distant d'au plus R_b, il existe donc potentiellement des cases captées par plusieurs capteurs.

On note $\lambda(R_c, R_b)$ le nombre minimal de cases présentes dans les boules de rayon R_c centrées en deux capteurs distants d'au plus R_b.

$$\lambda(R_c, R_b) := \min_{(x,y)\in\mathbb{Z}^2} \{\mathbf{card}(B_{R_c} \cap ((x, y) + B_{R_c})), x^2 + y^2 \leq R_b^2\}$$

Pour les valeurs qui nous intéressent, on trouve :

(R_c, R_b)	$(1,1)$	$(1,2)$	$(2,2)$	$(2,3)$	$(3,3)$	$(3,4)$
λ	2	1	5	2	12	7

Ainsi, pour chaque arête du graphe de communication, il existe au moins $\lambda(R_c, R_b)$ cases captées par plus de deux capteurs. Comme un graphe connexe à b possède au moins $b-1$ arêtes, on en déduit qu'il existe au moins $(b-1)\cdot\lambda(R_c, R_b)$ cases captées par deux capteurs. (Si une case est captée par n capteurs, elle est comptée $n-1$ fois).

On en déduit que

$$b \cdot K_{R_c} \geq N^2 + (b-1)\cdot\lambda(R_c, R_b)$$

c'est-à-dire,

$$b \geq \frac{R_c^2 - \lambda(R_c, R_b)}{K_{R_b} - \lambda(R_c, R_b)}$$

et comme le nombre de capteur est entier, on trouve :

$$b \geq \lceil \frac{N^2 - \lambda(R_c, R_b)}{K_{R_b} - \lambda(R_c, R_b)} \rceil$$

On a référencé les valeurs de ces bornes inférieures dans un tableau (figure 4.2) afin de les comparer aux valeurs que l'on a obtenu.

	$(1,1)$	$(1,2)$	$(2,2)$	$(2,3)$	$(3,3)$	$(3,4)$
10	33	25	12	9	6	5
15	80	56	28	21	13	10
20	133	100	50	37	23	18
30	310	225	112	82	53	41
40	533	400	200	146	94	73
50	833	625	312	228	147	114

Figure 1: Bornes inférieures théoriques.

Remarque. En se cassant un peu plus la tête, on peut affiner encore davantage ces bornes.

4.3 Résultats pratiques

Nous présentons dans cette partie les différents résultats que nous avons obtenu : Le tableau en figure4.3 présente les valeurs obtenues en moyenne et dans le meilleur cas pour différentes valeurs de N, R_c et R_b. On a aussi relevé le temps moyen de convergence dans chacun des cas.

	Couple (Rc,Rb)					
	Valeur moyenne / Valeur dans le meuilleur cas					
	Temps de convergence					
	(1,1)	(1,2)	(2,2)	(2,3)	(3,3)	(3,4)
N=10	**39/39**	**30/29**	**17/17**	**12.2/12**	**7.2/7**	**6.1/6**
	0,44	0,87	0,37	0,56	0,47	0,76
N=15	**83.1/81**	**66.1/64**	**37.4/33**	**27.1/26**	**16.2/15**	**13.2/13**
	6,8	7,6	3,2	4,9	4	6,7
N=20	**146.3/146**	**117.4/114**	**63.7/60**	**47/46**	**28.3/28**	**22.6/22**
	28,1	34,3	13,8	20,6	16,4	28,5
N=30	**327/322**	**255.6/254**	**140.4/137**	**104.2/102**	**64.4/63**	**50.8/50**
	334	373	112	127	96	165
N=50	**960.5/960**	**716/712**	**401.5/400**	**294.5/294**	**191/191**	**147.5/146**
	6880	4210	1480	1310	879	1260

Figure 2: Résultats obtenues par l'algorithme avec $S = 40$, superposition $= R_c + 5$, durée vie $= 5$

Le tableau de la figure 4.3 rescense les meilleurs résultats obtenues. Il peut être intéressant des les comparer aux bornes théoriques présentés à la figure 4.2.

	$(1,1)$	$(1,2)$	$(2,2)$	$(2,3)$	$(3,3)$	$(3,4)$
10	39	29	17	12	7	6
15	81	64	33	26	15	13
20	146	114	60	47	27	22
30	322	254	137	102	63	50
40	624	459	260	192	120	94
50	960	712	400	294	191	146

Figure 3: Meilleurs solutions obtenues avec l'approche génétique.

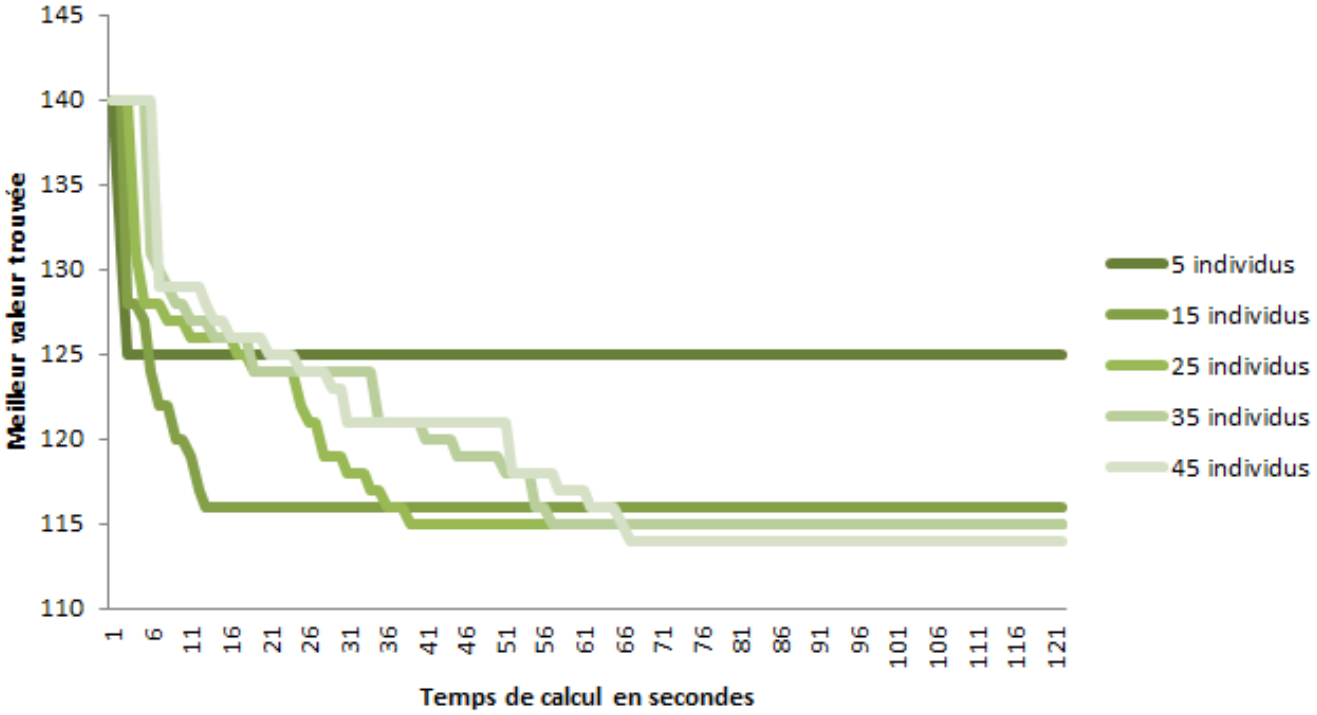


Figure 4: Influence de S sur la vitesse de convergence.

5 Choix des différents paramètres.

Nombre d'accouplements lors d'une génération. Ce choix est arbitraire et n'influence en rien la vitesse d'exécution, mise à part la fréquence à laquelle on met à jour le meilleur représentant. On pourrait même considérer qu'une génération est produit d'un seul accouplement de la précédente.

Nombre d'individu S. Ce paramètre influence grandement la vitesse et les l'optimalité du résultat renvoyé.

- Quand S est petit, l'initialisation et de calcul d'une génération est rapide mais le peu de diversité génétique conduit à des résultats généralement loin des optimaux, en particulier pour des grandes valeurs de N.
- A l'inverse, quand S est grand, le temps d'initialisation et de calcul d'une génération est important mais les recherches sont plus diversifiées et à long terme les résultats renvoyés sont bons. Ce pose aussi le problème de la complexité spatiale. En effet pour des grandes valeurs de N, la mémoire vive est saturé et il est donc difficile d'avoir de bonnes garanties d'optimalité.

Le graphique 5 illustre ces remarques pour $N = 20$ et $(R_c, R_b) = (1, 2)$.

Lorsque N est petit, il n'est pas judicieux de prendre S grand, même si l'on souhaite avoir des garanties d'optimalité. Par exemple, $S = 30$ est suffisant pour $N = 10$.

Durée maximale de stagnation. La durée maximale de stagnation ne doit pas être choisie trop petite, sinon l'algorithme risque de s'arrêter avant d'avoir vraiment exploiter ses possibilités. Mais ça ne sert à rien de le choisir trop grand, si l'on a atteint la valeur optimale, il ne sert à rien de continuer à calculer trop longtemps.

Degré de superposition lors de l'accouplement. Nous n'avons pas identifié de changements au niveau de la vitesse de convergence lorsque nous faisons varier ce paramètre. Il est préférable de choisir ce paramètre supérieur au rayon de captation pour garantir aux progénitures d'être viables. Cependant, expérimentalement, il semble sous pour certaines valeurs de N et de S, il est toujours plus intéressant de choisir ce paramètre grand, voire égal à la largeur de la grille. Inclure de façon "excessive" des mutations lors de la reproduction semble donc être une approche intéressante dans certains cas.

Durée de vie maximale d'une individu. Choisir une durée de vie maximale grande favorise le parcours en profondeur alors qu'à l'inverse, une durée de vie petite encourage la diversité des solutions rencontrée. Experimentalement, cette influence ne se remarque que sur des valeurs de N grandes et des valeurs de S petites (relativement à la taille de la grille). Dans ces cas il semble intéressant de favoriser une durée de vie courte (par exemple 5). Cependant, choisir une durée de vie trop petite (par exemple 1) serait une erreur.

6 Améliorations

Obtenir une meilleure génération initiale. En imaginant une autre approche gloutonne, favorisant par exemple les graphes peu denses et optimisant la couverture (un point doit être capté par le moins de capteurs possible) pourrait permettre a priori d'accelérer l'algorithme et obtenir de meilleures bornes.

Fonction d'évaluation pour la séléction. Ne pas s'intéresser seulement aux nombres de capteurs comme critère de selection mais également à des caractéristique structurelles comme la façon dont les capteurs sont couverts, et la faible densité du graphe de communication, pourrait a priori donner des résultats intéressants.

Test de connexité. A chaque fois qu'une grille est réduite, on fait appel à un test de connexité qui consiste à parcourir l'ensemble des sommets du graphe de communication. Or, ce test est très souvent appelé successivement. Il serait donc intéressant d'utiliser un algorithme qui détermine si un sommet est un isthme sans avoir à parcourir l'ensemble du graphe.

Programmation parallèle. Cette méta-heuristique est particulièrement adaptée à une implémentation utilisant plusieurs cœurs en parallèles car le partage du travail est particulièrement simple à mettre en place :

- Pour l'initialisation, il suffit d'appeler l'heuristique gloutonne sur chaque cœur en parallèle. Si l'on dispose de κ cœurs, le temps de calcul de la première génération est divisé par κ.
- Pour le calcul des générations successives, chaque famille (deux parents et deux enfants) peut être calculée séparément. Le temps de calcul est donc divisé par $\min(\kappa, \frac{s}{4})$.

Étude plus fine des paramètres. Il serait intéressant d'étudier plus en détail l'influence le dégré de superposition et la durée de vie maximale en fonction des valeurs de N et de S. Une meilleure compréhension de ces paramètres permettrait a fortiri d'accélerer la vitesse de l'algorithme.

7 Conclusion

La résolution du problème de couverture connexe ne peut en général, être faite de manière exacte en temps raisonnable. Néanmoins ce projet à permis de mettre en place une heuristique approchant efficacement la solution du problème en utilisant des méthodes de types évolutionnaire. Malgré des calculs coûteux en espace mémoire, l'heuristique a fait ses preuves en terme d'efficacité. Un bon ajustement des paramètres de l'heuristique (nombre d'individus, type de croisements, type de mutations et critère de sélection) permet d'exploiter les bienfaits du principe de sélection naturelle.