Recherche à voisinage variable et programmation semidéfinie positive dans les réseaux de Troisième Génération (3G)

R. Lopez[1] and A. Lisser[2]

[1] Laboratoire de Recherche en Informatique,
Bat 490 Université Paris-Sud, 91405 Orsay Cedex
lopez@lri.fr
[2] Laboratoire de Recherche en Informatique,
Bat 490 Université Paris-Sud, 91405 Orsay Cedex
lisser@lri.fr

Résumé Face à l'émergence des réseaux de téléphonie mobile, les opérateurs doivent mettre en place des structures de plus en plus perfectionnées pour satisfaire les attentes des consommateurs. En particulier, ces dernières années se développent les réseaux dits de troisième génération, capables non seulement de transporter la voix, mais également du contenu multimédia tel que de la musique, des pages web et de la vidéo. Pour cela, plusieurs normes ont été développées, comme par exemple l'UMTS. Une étape importante dans le transport des données est la réception au niveau des tours des signaux transmis par les terminaux mobiles. En effet, la tour reçoit un signal composite formé de l'ensemble des signaux des utilisateurs présents dans la cellule ; il faut donc décomposer ce signal pour retrouver chaque signal individuel (démultiplexage), et ce de manière aussi fidèle que possible. Compte tenu des hypothèses du modèle de transmission et de réception, cette tâche est équivalente mathématiquement au problème de MAX CUT en variables binaires, et donc NP-difficile. Pour réaliser ce démultiplexage, l'approche la plus simple est de faire une recherche exhaustive parmi les signaux possibles et de garder celui qui maximise la vraisemblance. Ce détecteur est optimal en résultat, mais demande un temps de calcul exponentiel (deux puissance le nombre d'utilisateurs cas à vérifier). Comme un tel temps de calcul n'est pas utilisable pour du traitement de signal, on fait appel à d'autres méthodes de résolution. Nous utilisons ici la relaxation Semidéfinie Positive (SDP) pour obtenir une borne inférieure (relaxation continue) de l'optimum, que nous passons ensuite à un algorithme de recherche à voisinage variable (VNS) pour trouver la meilleure solution possible (borne supérieure). Nous montrons que cette approche permet d'obtenir des résultats aussi bons que ceux trouvés jusqu'à présent sur les petites instances, et que celle-ci permet également de travailler sur des instances plus grandes que celles atteintes jusqu'à présent.

Mots-Clefs. CDMA ; Détection multi-utilisateurs ; Programmation semidéfinie positive.

1 Introduction

Ces dernières années ont vu l'émergence des systèmes de communications sans fil, en particulier la téléphonie mobile personnelle. Les premières technologies ont rapidement montré leur limites : les appareils étaient assez gros, lourds, et ne disposaient que d'une autonomie limitée. Les améliorations apportées ont grandement changé cela, l'autonomie passant de environ 2 heures en communication à plus de 10 heures, voire plus pour les appareils les plus performants. Ce progrès est en partie dû à l'évolution du système de communication, qui, plus efficace, nécessite moins de puissance pour obtenir une qualité égale. Cela s'est également accompagné d'une augmentation des capacités d'accueil (en termes d'utilisateurs simultanés) et de la demande des utilisateurs (le nombre d'abonnés a augmenté de manière très soutenue ces dix dernières années), et a donné lieu à des problèmes d'optimisation largement étudiés (localisation des bornes relais, affectation des fréquences entre les cellules...). Les évolutions les plus récentes ont poussé encore plus loin les capacités des réseaux (débit plus élevé), apportant en plus du transport de la voix et des messages textes la possibilité d'échanger des images, des jeux et de la vidéo (visioconférence, streaming).
Pour permettre ces nouveaux services, il faut une très bonne qualité du signal, et donc très peu d'erreurs entre ce qui est émis et reçu. Pour cela, les réseaux de Troisième Génération (3G, qui repose sur différentes interfaces radio, UMTS[3] ou CDMA2000[4]) utilisent une méthode de codage visant à diminuer le risque d'erreur de transmission. Une étape importante est au moment de la réception du signal par la base, lorsque les signaux de chacun des utilisateurs doivent être reconnus. En effet, ceux-ci émettent simultanément, et leur signal arrive sous forme agrégée à la tour, qui doit pouvoir identifier chaque signal. Pour cela, plusieurs méthodes de modulation/démodulation existent.
L'étape de démodulation, puis de détection de chaque signal est donc cruciale pour maintenir la qualité de la communication. Cette étape est également complexe, car le signal n'arrive pas dans un état parfait : il y a des interférences (du "bruit"), les utilisateurs sont souvent en mouvement et les signaux se répercutent sur les bâtiments (ce qui produit de l'effet Doppler), les signaux perdent en puissance avec la distance parcourue (affaiblissement, ou *fading*) et enfin, tous les utilisateurs n'émettent pas de manière synchrone. Le détecteur a donc un travail complexe à accomplir ; il a été montré [12] que le détecteur optimal dans le cas asynchrone est un détecteur de séquences par maximum de vraisemblance, où l'on détecte les signaux de tous les utilisateurs de manière simultanée, et que le programme d'optimisation associé à ce détecteur optimal est réductible à un problème de MAX CUT (NP-dur).
Une possibilité est de relâcher certaines contraintes pour obtenir un problème solvable non linéaire, que l'on peut alors approximer, ou borner. Plusieurs méthodes existent alors pour trouver une bonne (pas nécessairement la meilleure) solution au problème initial. Nous nous sommes intéressés à deux différentes mé-

[3] UMTS : *Universal Mobile Telecom. Services*
[4] CDMA : *Carrier Division Multiple Access*

thodes : d'une part par la relaxation semidéfinie positive [14], et d'autre part par une métaheuristique : la recherche à voisinage variable [3]. Le développement de ce document est le suivant : dans une première partie, nous présentons le modèle de communication étudié, et sa formalisation mathématique ; nous développons ensuite les principes de la programmation semidéfinie positive, puis dans la partie suivante ceux de la recherche à voisinage variable. Enfin, nous avons réalisé des simulations afin de comparer l'efficacité de ces deux méthodes, tant en terme de qualité que de complexité en temps de calcul.
Les articles de référence dans l'application de ces méthodes sont les travaux de Tan et Rasmussen [10], et un certain nombre d'articles faisant suite développant d'autres métaheuristiques (recherche tabou, algorithmes génétiques) ou faisant suite aux travaux de Tan et Rasmussen sur l'utilisation de la programmation semidéfinie positive pour détecter les symboles[5]. A notre connaissance, il n'y a pas d'articles utilisant d'une part la recherche à voisinage variable ([10] se limite à une recherche locale itérative), et simultanément utilisant la programmation semidéfinie pour améliorer les performances de l'algorithme de recherche à voisinage variable. Les travaux que nous avons réalisés à ce jour nous ont permis de trouver de bonnes bornes de la fonction à optimiser, notamment la borne supérieure qui s'est révélée être dans de nombreux cas confondue avec le résultat optimal, et ce avec un gain important en terme de temps de calcul.

2 Modélisation du système CDMA

Nous nous intéressons ici à la modélisation d'un canal CDMA, que nous allons présenter et décrire. Ensuite nous présenterons les principes de la détection multi-utilisateurs dans ce canal et les critères de décision qui y sont liés.

2.1 Modélisation du système CDMA

On se base sur un canal CDMA partagé par K utilisateurs simultanés, chacun se voyant attribuer une signature $p_k(t)$ de durée T, où T est l'intervalle du symbole. On peut exprimer la signature (code d'étalement) de la manière suivante :

$$p_k(t) = \sum_{n=0}^{N-1} a_k(n) p(t - nT_c) \tag{1}$$

où $\{a_k(n), 0 \leq n \leq N-1\}$ est une séquence de code composée de N chips qui prennent pour valeur $\{\pm 1\}$ (de manière équivalente, on peut noter $a_k(n) = (-1)^{c_{k,n}}$, avec $c_{k,n}$ une séquence binaire de longueur N ; voir en annexe 1 pour un exemple de génération de séquence utilisant la méthode des matrices de Hadamard), et $p(t)$ est une impulsion de durée T_c, où T_c est le temps de chip. On a donc N chips par symbole et $T = NT_c$. On suppose que l'énergie utilisée pour chacune des K signatures est normalisée à un (c'est-à-dire que $\|p_k\|^2 = \int_0^T p_k^2(t)dt = 1$).

[5] Données transmises. On les suppose codées sous forme binaire $(-1, 1)$.

Les impulsions $p(t)$ (signal d'horloge) peuvent prendre plusieurs formes, par exemple de type rectangulaire (couramment utilisée) :

$$p(t) = \begin{cases} 1, & \text{si } 0 \leq t < T_c, \\ 0, & \text{sinon} \end{cases}$$

ou bien de type sinusoïdale (plus rare) :

$$p(t) = sinc\left(\frac{2t}{T_c} - 1\right).$$

On dénote la séquence des données transmises par l'utilisateur K par $\{d_k(m)\}$, la valeur de chaque symbole pouvant être choisie de manière uniforme dans l'ensemble $\mathcal{D}$ des symboles possibles. Toutes les séquences de données sont équiprobables et chaque symbole est statistiquement indépendant des autres symboles, ainsi qu'entre les différents utilisateurs. On s'intéresse à une séquence de longueur arbitraire L. Le signal faible amplitude (*low-pass*) reçu sur le canal pour l'utilisateur k peut alors s'exprimer par :

$$s_k(t) = c_k(t) * \left[\sum_{i=1}^{L} d_k(i) p_k(t - iT)\right] \tag{2}$$

pour $0 \leq t \leq (L+1)T$, où * est le produit de convolution, et $c_k(t)$ la réponse impulsionnelle du canal (dans $\mathbb{C}$). Le signal composé transmis par les K utilisateurs peut s'écrire

$$s(t) = \sum_{k=1}^{K} \left[c_k(t) * \sum_{i=1}^{L} d_k(i) p_k(t - iT - \tau_k)\right] \tag{3}$$

avec $\{\tau_k\}$ les délais de transmissions, qui satisfont la condition $0 \leq \tau_k \leq T$ pour $k = 1, 2, \ldots, K$. Sans perte de généralité, on pose $0 \leq \tau_1 \leq \tau_2 \leq \cdots \leq \tau_k \leq T$ (on ordonne les utilisateurs par délai d'arrivée du signal, i.e. le premier utilisateur est celui dont le signal arrive le premier). Il s'agit du modèle pour un signal multi-utilisateurs transmis de manière asynchrone. Dans le cas d'un système synchrone, on a $\tau_k = 0$ pour $k = 1, 2, \ldots, K$.

Pour un canal non sélectif de fréquence, la bande passante du signal est sensiblement plus faible que la bande passante de cohérence du canal, et les composantes multi trajets ne sont alors pas solubles [9]. Dans ce cas, le signal reçu est le signal transmis multiplié par un processus aléatoire à valeurs complexes qui représente les caractéristiques temporelles variantes du canal. De plus, si on suppose que la durée du signal est significativement plus courte que le temps de cohérence du canal, alors le canal s'atténue progressivement et les paramètres du canal, l'atténuation et le décalage de phase, sont essentiellement constants pendant la durée d'un intervalle de symbole [9]. Si l'on applique ces hypothèses à tous les utilisateurs, alors ceux-ci rencontrent le même canal AWGN (Additive White Gaussian Noise, bruit blanc additif gaussien). Le signal transmis est alors

supposé corrompu par du bruit blanc additif gaussien. En conséquence, le signal reçu peut s'exprimer par

$$r(t) = s(t) + n(t) \tag{4}$$

où $n(t)$ est le bruit, de variance σ^2.

2.2 Détection des signaux avec du bruit blanc gaussien

La détection des signaux peut être modélisée comme un modèle de test de M hypothèses, où M est le nombre possible de combinaisons de symboles de données $\mathbf{d} \in \mathcal{D}^{LK}$. Chaque combinaison $\mathbf{d}$ dans l'hypothèse H_i est notée $\mathbf{d}_i$. Le test d'hypothèse peut se modéliser par :

$$\begin{aligned} H_i : r(t) = s(t, \mathbf{d}_i) + n(t), \qquad & 0 < t < T, \\ & \mathbf{d}_i \in \mathcal{D}^{LK}, 1 \leq i \leq M \end{aligned} \tag{5}$$

avec $s(t, \mathbf{d}_i)$ le signal sous l'hypothèse H_i, avec pour symboles de données $\mathbf{d}_i$, et $n(t)$ le bruit blanc gaussien additif.
Le problème est d'observer $r(t)$ et de décider quelle hypothèse est vraie avec une probabilité d'erreur minimale. On observe $r(t)$, une onde aléatoire en temps continu. La première étape est de la réduire en un ensemble de variables aléatoires regroupées dans un vecteur dit vecteur reçu. Une méthode générale pour obtenir le vecteur reçu $\mathbf{r}$ est par développement en séries [11]. On a :

$$r(t) = s(t) + n(t) \tag{6}$$

On développe alors les trois composantes sous forme d'un ensemble de fonctions orthonormales, dont on déduit :

$$\begin{aligned} r_k &= \int_0^T r(t)\phi_k^*(t)\, dt \\ &= \int_0^T \left[s(t) + n(t)\right] \phi_k^*(t)\, dt \\ &= \int_0^T s(t)\phi_k^*(t)\, dt + \int_0^T n(t)\phi_k^*(t)\, dt \\ &= s_k + n_k \end{aligned} \tag{7}$$

où r_k est le k-ième élément du vecteur $\mathbf{r}$ et $\phi_k^*(t)$ la k-ième fonction de base, orthonormale, obtenue par exemple par la méthode de Gram-Schmidt [9]. De manière similaire à $\mathbf{r}$, le vecteur $\mathbf{s}$ composé des éléments $\{s_k\}$ est appelé le vecteur de signal et $\mathbf{n}$ le vecteur de bruit. D'après le théorème de la non-pertinence[6] [15], on a seulement besoin de réduire $s(t)$ en un ensemble discret de variables. Comme $n(t)$ est supposé statistiquement indépendant de $s(t)$, la partie de $n(t)$ hors du champ couvert par la représentation en séries de $s(t)$ n'a pas d'effet

[6] *Un récepteur optimal peut ignorer un vecteur* $\mathbf{r}_2$ *si et seulement si* $p_{\mathbf{r}_2|\mathbf{r}_1,\mathbf{s}} = p_{\mathbf{r}_2|\mathbf{r}_1}$.

sur la détection. Le théorème de la réversibilité[7] [15] suppose que l'opération de représentation en séries soit réversible (c'est-à-dire que l'on peut exactement retrouver l'input) pour que la statistique soit suffisante.

Une statistique suffisante est donc l'output du filtre adapté (*Matched Filter*, noté MF), que l'on note

$$\begin{aligned} y_k(i) &= \int_{iT+\tau_k}^{(i+1)T+\tau_k} r(t)p_k(t-iT-\tau_k)\,dt \\ &= \int_{iT+\tau_k}^{(i+1)T+\tau_k} s(t)p_k(t-iT-\tau_k) + n(t)p_k(t-iT-\tau_k)\,dt \end{aligned} \tag{8}$$

Sous forme vectorielle, on obtient [12] :

$$\mathbf{y} = \mathbf{RCd} + \mathbf{n} \tag{9}$$

Avec $\mathbf{y}$ le vecteur des séquences de bits sortant du filtre adapté, $\mathbf{d}$ le vecteur des séquences de bits envoyés et $\mathbf{n}$ le vecteur des séquences de bruit sur le canal. $\mathbf{C}$ est une matrice diagonale contenant les coefficients de canaux des utilisateurs. $\mathbf{R}$ est la matrice des corrélations croisées, de dimensions $KL \times KL$. Le vecteur $\mathbf{n}$ de bruit gaussien a une moyenne nulle et pour matrice d'autocorrélation (avec $\sigma^2 = N_0/2$)

$$E\left[\mathbf{n}\mathbf{n}^H\right] = \sigma^2\mathbf{R} \tag{10}$$

Le détecteur optimal sur le maximum de vraisemblance (MV) choisit une hypothèse $\hat{\mathbf{d}}$ de MV, compte tenu du résultat du MF, et suppose une connaissance parfaite de $\mathbf{C}$ et de $\mathbf{R}$:

$$\hat{\mathbf{d}} = \arg\max_{\mathbf{d}\in\mathcal{D}} p(\mathbf{y}\,|\,\mathbf{d}). \tag{11}$$

On travaille dans un canal AWGN, la fonction de logvraisemblance basée sur $p(\mathbf{y}\,|\,\mathbf{d})$ peut s'écrire

$$F(\mathbf{d}) = 2\mathrm{Re}\{\mathbf{y}^H\mathbf{Cd}\} - \mathbf{d}^H\mathbf{C}^H\mathbf{RCd}.$$

En effet, comme $\mathbf{n}$ est gaussien et que $E(\mathbf{y}|\mathbf{d}) = \mathbf{RCd}$, on a

$$p(\mathbf{y}|\mathbf{d}) = \frac{1}{(\pi)^K\|\sigma^2\mathbf{R}\|} \times \exp\left[-(\mathbf{r}-\mathbf{RCd})^H(\sigma^2\mathbf{R})^{-1}(\mathbf{r}-\mathbf{RCd})\right] \tag{12}$$

Le problème du maximum de vraisemblance s'écrit alors :

$$\begin{aligned} \hat{\mathbf{d}} &= \arg\max_{\mathbf{d}\in\mathcal{D}^{KL}} \frac{1}{(\pi)^K\|\sigma^2\mathbf{R}\|} \times \exp\left[-(\mathbf{y}-\mathbf{RCd})^H(\sigma^2\mathbf{R})^{-1}(\mathbf{y}-\mathbf{RCd})\right] \\ \hat{\mathbf{d}} &= \arg\max_{\mathbf{d}\in\mathcal{D}^{KL}} \exp\left[-(\mathbf{y}-\mathbf{RCd})^H\mathbf{R}^{-1}(\mathbf{y}-\mathbf{RCd})\right] \\ \hat{\mathbf{d}} &= \arg\min_{\mathbf{d}\in\mathcal{D}^{KL}} \mathbf{d}^H\mathbf{C}^H\mathbf{RCd} - 2\mathrm{Re}\{\mathbf{y}^H\mathbf{Cd}\} \end{aligned} \tag{13}$$

[7] Ce théorème nous dit que *la probabilité minimale d'erreur atteignable n'est pas affectée par l'introduction d'une opération réversible sur l'output du canal.* Une opération est réversible si l'information d'origine peut être exactement récupérée depuis celle ayant subi l'opération. C'est un corollaire du théorème de la non-pertinence.

Pour mettre en avant les détails de l'application de la programmation semi-définie positive, nous considérerons uniquement le CDMA synchrone. Dans le cas de communications synchrones, on a $\tau_k = 0$, et chaque utilisateur interférant produit exactement un symbole qui interfère avec le symbole attendu. Dans le cas d'un canal mono-trajet AWGN, il est suffisant de s'intéresser au signal reçu au cours d'un intervalle. Quand les coefficients de canal sont réels et les symboles $\mathbf{d}$ sont binaires, i.e., $\mathcal{D} \in \{-1, 1\}^K$, le problème d'optimisation précédent devient

$$\hat{\mathbf{d}} = \arg \min_{\mathbf{d} \in \{\pm 1\}^K} \mathbf{d}^T \mathbf{C}^T \mathbf{R} \mathbf{C} \mathbf{d} - 2\mathbf{y}^T \mathbf{C} \mathbf{d} \tag{14}$$

Pour simplifier l'écriture et la compréhension de la technique de la programmation semidéfinie, nous nous concentrerons sur le cas d'un système synchrone sur des canaux réels avec une modulation binaire. Il est possible, au prix d'une notation complexe, de l'étendre au cas de la modulation QPSK.

3 Programmation SDP et détection multi-utilisateurs

Le choix d'utiliser la relaxation SDP vient des résultats obtenus pour les problèmes de type MAX-CUT, en particulier ceux de Goemans et Williamson [2] (et Feige et Goemans [1]). La relaxation SDP permet en effet d'obtenir de bonnes bornes (et par conséquent, une bonne approximation de la solution optimale par l'algorithme de Goemans et Williamson, ou de Zwick [16]). Pour plus de détails, voir la survey de H Wolcowitz [13]. On se base sur la formulation canal réel - symbole binaire, en posant $\mathbf{Q} = \mathbf{C}^T \mathbf{R} \mathbf{C}$, $\mathbf{c} = \mathbf{C}\mathbf{y}$ et $\mathbf{u} = \hat{\mathbf{d}}$. On obtient alors :

$$\begin{aligned} \mathbf{u} &= \arg \min_{\mathbf{u}} \mathbf{u}^T \mathbf{Q} \mathbf{u} - 2\mathbf{c}^T \mathbf{u} \\ &\text{s.c. } \mathbf{u} \in \{-1, 1\}^{n-1} \end{aligned} \tag{15}$$

avec $K = n - 1$. Les résultats suivants sont vrais pour n'importe quelle matrice $\mathbf{Q} = \mathbf{Q}^T$ [6]. En ajoutant une variable muette redondante u_n, on peut exprimer le programme précédent sous la forme :

$$\begin{aligned} [\mathbf{u}^*, u_n^*] = \arg \min_{[\mathbf{u}, u_n]} & \begin{bmatrix} \mathbf{u}^T & u_n \end{bmatrix} \begin{bmatrix} \mathbf{Q} & -\mathbf{c} \\ -\mathbf{c}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ u_n \end{bmatrix} \\ \text{s.c. } & \begin{bmatrix} \mathbf{u} \\ u_n \end{bmatrix} \in \{-1, 1\}^n, \quad u_n = 1. \end{aligned} \tag{16}$$

Comme la fonction de coût est symétrique, on n'a pas besoin de maintenir explicitement que $u_n = 1$, on pose alors

$$\mathbf{L} = \begin{bmatrix} \mathbf{Q} & -\mathbf{c} \\ -\mathbf{c}^T & 0 \end{bmatrix} \quad \text{et} \quad \mathbf{x} = \begin{bmatrix} \mathbf{u}^T & u_n \end{bmatrix}^T \tag{17}$$

on reformule alors le problème (15), qui devient

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \mathbf{x}^T \mathbf{L} \mathbf{x} \quad \text{s.c. } \mathbf{x} \in \{-1, 1\}^n. \tag{18}$$

On réécrit la fonction objectif sous la forme :

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \mathrm{tr}\{\mathbf{L}\mathbf{x}\mathbf{x}^T\}. \tag{19}$$

Pour tout $\mathbf{x} \in \{-1, 1\}^n$, la matrice $\mathbf{x}\mathbf{x}^T$ est semidéfinie positive, ses éléments diagonaux sont égaux à 1, et le rang de cette matrice est 1. On pose $\mathbf{X} = \mathbf{x}\mathbf{x}^T$, telle que $\mathbf{X}$ satisfait ces trois propriétés. Cette écriture est équivalente à :

$$\mathbf{X} = \begin{bmatrix} \mathbf{u}\mathbf{u}^T & \mathbf{u} \\ \mathbf{u}^T & 1 \end{bmatrix} \tag{20}$$

Le programme (18) devient alors :

$$\begin{aligned} &\mathbf{X}_1^* = \arg\min_{\mathbf{X}} \mathrm{tr}\{\mathbf{L}\mathbf{X}\} \\ &\text{s.c.} \quad \mathrm{diag}(\mathbf{X}) = \mathbf{e}_n, \quad \mathrm{rang}(\mathbf{X}) = 1, \quad \mathbf{X} \succeq 0. \end{aligned} \tag{21}$$

où $\mathrm{diag}(\mathbf{X})$ est le vecteur des éléments diagonaux de $\mathbf{X}$ et $\mathbf{e}_n$ le vecteur unitaire de dimension n. $\mathbf{X} \succeq 0$ signifie que $\mathbf{X}$ est semidéfinie positive. L'indice de $\mathbf{X}_1^*$ se rapporte à la contrainte de rang 1. L'équivalence entre le programme (18) et le programme (21) est montrée de manière générale dans le Lemme 3.1 de [7]. Si l'on relâche la contrainte de rang, on obtient alors le programme semidéfini suivant :

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \mathrm{tr}\{\mathbf{L}\mathbf{X}\} \quad \text{s.c. } \mathrm{diag}(\mathbf{X}) = \mathbf{e}_n, \quad \mathbf{X} \succeq 0. \tag{22}$$

avec $\mathrm{diag}(\mathbf{X})$ le vecteur des éléments diagonaux de $\mathbf{X}$.

Helmberg a développé un logiciel, `SBmethod` mettant en oeuvre la méthode du *Spectral Bundle*, que nous avons utilisé pour résoudre nos problèmes d'optimisation. La matrice de coût est construite sous Matlab, puis passée à `SBmethod` qui renvoie la matrice $\mathbf{X}^*$ des solutions approchées. Celle-ci nous permet alors de déduire un vecteur $\mathbf{u}^*$ de bits reçus par la base. On utilise ce logiciel car celui-ci est adapté à la manipulation de grandes matrices d'inconnues, converge rapidement pour ces grandes matrices, et exploite notre connaissance à l'avance de la trace de la matrice solution ($\mathrm{tr}(\mathbf{X}) = n$). Par ailleurs, dans la perspective d'utilisation de nos résultats dans de problèmes à très grande échelle (en dehors du champ de la détection multi-utilisateur), alors la méthode des faisceaux permet de travailler sur des problèmes plus importants que les outils basés sur la méthode des points intérieurs.

4 Recherche à voisinage variable

Dans cette partie, nous présenterons les principes de la VNS (pour plus de détails, se référer aux travaux de P. Hansen [3]), pour aboutir à nos algorithmes VNS et VNS + SDP.

4.1 Présentation

La recherche à voisinage variable (RVV, ou VNS en anglais) est une métaheuristique récente ([3], [4], [5], [8]) qui exploite l'idée de changement de voisinage tant lors de la descente vers des optima locaux que pour sortir des vallées qui les contiennent. On définit nos voisinages $N_k(\mathbf{x})$ comme les vecteurs $\mathbf{x}'$ se trouvant à distance k de $\mathbf{x}$. La distance est le nombre de bits de différence entre le vecteur $\mathbf{x}$ et le vecteur $\mathbf{x}'$.

La VNS est une métaheuristique qui permet de trouver une borne supérieure de la fonction à minimiser. En comparant cette borne avec celle trouvée par la méthode SDP, on peut voir si on obtient une bonne approximation de la solution optimale. L'heuristique VNS utilise les observations suivantes :
Obs 1 : Un minimum local dans une structure de voisinage donnée n'en est pas nécessairement un pour une autre.
Obs 2 : Un minimum global est un minimum local pour toutes les structures possibles de voisinages.
Obs 3 : Pour de nombreux problèmes, les minima locaux dans une ou plusieurs structures de voisinages sont relativement proches.

4.2 Recherche à voisinage variable

On peut combiner une recherche locale avec un changement de voisinage variable, ce qui nous donne notre algorithme VNS :

Initialisation. On choisit un ensemble de structures de voisinages N_k, avec $k = 1, \ldots, k_{max}$, que l'on utilisera lors de la phase de secousse, un ensemble de structures de voisinages N_l avec $l = 1, \ldots l_{max}$ utilisés lors de la recherche locale ; on trouve une solution initiale $\mathbf{x}$ que l'on améliore par RVNS ; on choisit une condition d'arrêt ; Dans notre situation, on pose $k_{max} = 1$ et $l_{max} = K$, et notre solution initiale est un vecteur généré aléatoirement.

Répéter la séquence suivante jusqu'à ce que l'on atteigne la condition d'arrêt (dans notre cas, un nombre d'itérations N_{iter}) :

(1) $k \leftarrow 1$

(2) *Répéter* les étapes suivantes jusqu'à ce que $k = k_{max}$ (on ne fait donc cette boucle qu'une fois) :

(a) *Secousse* : on génère un point $\mathbf{x}'$ au hasard dans le k-ième voisinage de $\mathbf{x}$ ($\mathbf{x}' \in N_k(\mathbf{x})$) (le voisinage immédiat).

(b) *Recherche locale par VND* :

(b1) $l \leftarrow 1$

(b2) *Répéter* les étapes suivantes jusqu'à ce que $l = l_{max}$:

- Exploration du voisinage : On cherche le meilleur voisin $\mathbf{x}''$ de $\mathbf{x}'$ dans $N_l(\mathbf{x}')$;

- Se déplacer ou non : Si $f(\mathbf{x}'') < f(\mathbf{x}')$ alors $\mathbf{x}' \leftarrow \mathbf{x}''$ et $l \leftarrow 1$; sinon $l \leftarrow l + 1$ (ici $f(\mathbf{u}) = \mathbf{u}^T\mathbf{Q}\mathbf{u} - 2\mathbf{c}^T\mathbf{u}$; c.f. (15)) ;

(c) *Se déplacer ou non* : Si l'optimum local $\mathbf{x}''$ est meilleur que le point d'origine $\mathbf{x}$, alors on s'y déplace ($\mathbf{x} \leftarrow \mathbf{x}''$), et on reprend la recherche dans N_1 ($k \leftarrow 1$) ; sinon on incrémente k ($k \leftarrow k + 1$).

Répéter N_{iter} fois la procédure permet de repartir à partir du meilleur point trouvé, éventuellement dans différentes directions si le point est dans une vallée "profonde". L'algorithme nous renvoie un vecteur $\mathbf{x}$ de bits reçus par la base.

4.3 VNS + SDP

Une des étapes de la recherche VNS, le choix du point initial, se fait habituellement de manière aléatoire, ce qui peut dans certains cas donner une convergence rapide, mais peut également entraîner une perte de temps si l'on a tiré un point situé dans une vallée éloignée de l'optimum recherché. Cela a pour effet de rendre le temps de recherche sujet à des variations imprévisibles de temps de calcul. Une solution pour remédier à cet effet est de choisir une solution initiale par une autre méthode. Ainsi, nous avons utilisé la méthode SDP pour rechercher une solution approchée que nous arrondissons ensuite (arrondi simple[8]) ($x^* = \text{sign}(\mathbf{u})$), à partir de laquelle la recherche VNS va opérer. Il faut noter que l'on ne fait pas une résolution complète en SDP, mais que l'on limite le temps de calcul (les paramètres de `SBmethod` sont "-te 0.001 -tt 1.2", qui limitent la précision à 1.10^{-3} et le temps d'exécution à 1.2 secondes). L'algorithme obtenu est alors quasiment identique au précédent, à la différence que la valeur initiale n'est plus générée aléatoirement mais obtenue par la méthode SDP. Un effet intéressant est que cela diminue le nombre d'itérations N_{iter} nécessaire pour converger vers une solution optimale.

5 Simulations

Nous nous intéressons dans cette section au BER (Bit Error Rate), qui, du point de vue de l'usager, est l'aspect principal, et nous permet de mesurer la performance de nos différentes méthodes de détection. Le BER se définit comme le rapport du nombre d'erreurs sur le nombre de bits transmis.

$$BER = \frac{\text{Nombre d'erreurs}}{\text{Nombre de bits transmis}} \tag{23}$$

Nous avons simulé différents scénarios : un premier avec $K = 10$ et $N = 32$, un cas un peu plus chargé avec $K = 24$ et $N = 32$, un cas plus délicat, avec $K = 48$ et $N = 64$ puis $N = 96$, et enfin un cas très chargé, avec $K = 60$ et $N = 64$ puis $N = 96$. Les deux premiers cas nous permettent de comparer les résultats obtenus par l'algorithme VNS avec ceux obtenus par recherche exhaustive[9] (noté ML),

[8] Nos résultats expérimentaux montrent que l'arrondi simple et un algorithme plus complexe, tel que celui de Goemans et Williamson nous donnent une qualité similaire en terme de log-vraisemblance ; on utilise alors celui le moins coûteux en temps de calcul

[9] Pour le cas $K = 24$, la recherche purement exhaustive est limitée au cas où $SNR/dB \leqslant 6$, l'évaluation requérant un temps de calcul trop important (près de 1000 secondes par simulation) pour obtenir une précision statistiquement suffisante (jusqu'à 4000 simulations sont nécessaires) ; dans le cas où $SNR/dB > 6$, on a donc procédé par approximation asymptotique

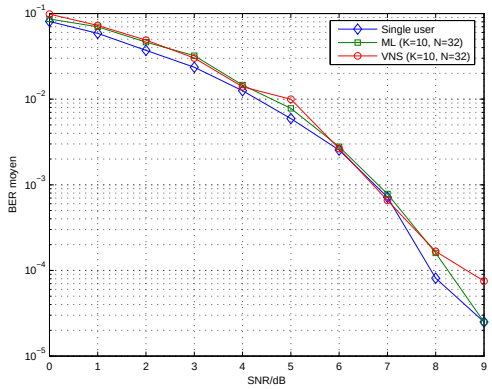

Fig. 1. Taux moyen d'erreur dans le cas peu chargé ($K = 10$)

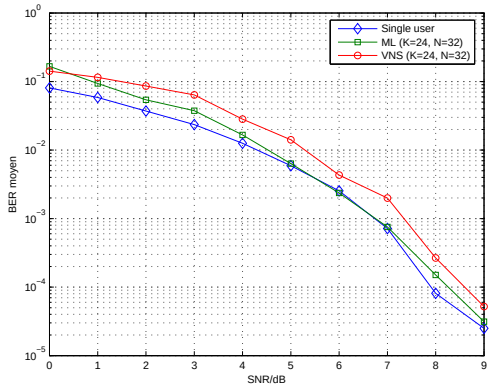

Fig. 2. Taux moyen d'erreur dans le cas moyennement chargé ($K = 24$)

et le cas de l'utilisateur unique [Fig. 1 et 2]. Les cas suivants nous permettent de mesurer les performances dans des situations plus complexes non étudiées jusqu'à présent [Fig. 3 et 4]. Il apparaît que pour les cas traités par les travaux antérieurs, on obtient des résultats comparables en terme de taux d'erreur. Ces performances sont proches des résultats idéaux du cas où un utilisateur unique serait présent, et de ceux obtenus par recherche exhaustive. Pour les cas plus chargés, nos algorithmes (VNS et VNS+SDP (noté VNS+)) continuent d'obtenir de bons résultats. On note toutefois que dans le cas où la capacité du canal est quasiment saturée, le VNS seul subit une forte dégradation de ses performances [Fig 4] alors que le VNS+ reste performant. La désaturation (augmentation de la capacité de $N = 64$ à $N = 96$) rétablit la performance du VNS.

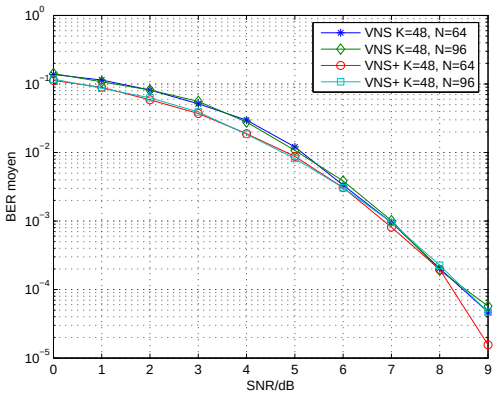

Fig. 3. Taux moyen d'erreur dans le cas plus chargé ($K = 48$)

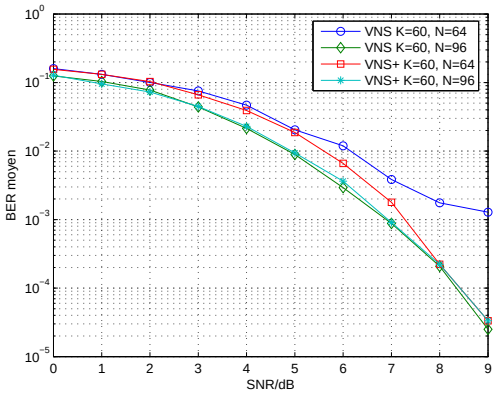

Fig. 4. Taux moyen d'erreur dans le cas plus chargé ($K = 60$)

Le temps de calcul de la recherche exhaustive est exponentiel avec le nombre d'utilisateurs ($O(2^K)$) [Fig. 5], dès lors que le nombre d'utilisateurs est suffisamment important ($K = 13$), le temps de calcul par la méthode SDP ou VNS est inférieur, dès que K est grand ($K > 16$), l'écart devient significatif (VNS et SDP prennent alors moins de 10% du temps nécessaire à la recherche exhaustive). Une deuxième série d'évaluation du temps de calcul a été réalisée pour le cas où $N = 64$ [Fig. 6]. Celle-ci montre que les performances en temps de calcul du VNS+ sont légèrement meilleures que le VNS seul, dès lors que $K > 12$, et que cet écart s'amplifie dans les cas les plus chargés. Ces résultats sont intéressants si on les rapproche de ceux concernant le BER. En effet, à temps de calcul inférieur, le VNS+ obtient des résultats bien meilleurs que le VNS dans les cas

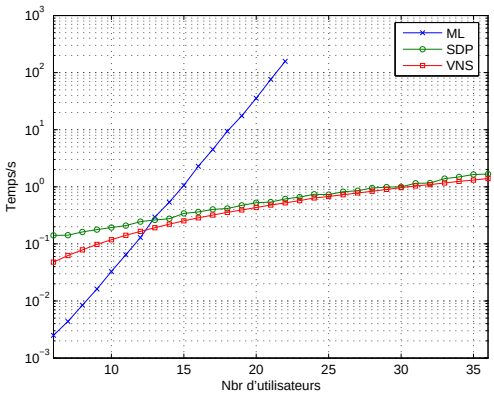

Fig. 5. Temps moyen de calcul par instance ($N = 32$)

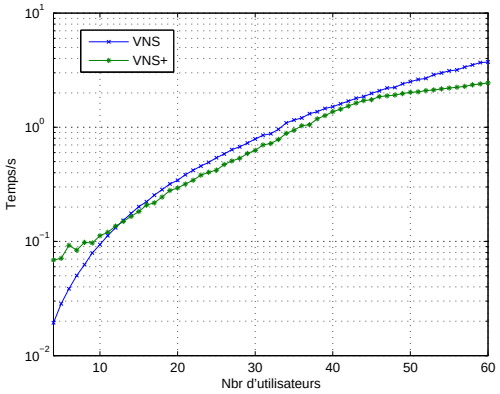

Fig. 6. Temps moyen de calcul par instance ($N = 64$)

quasi-saturés. Les simulations ont été exécutées dans l'environnement MATLAB sur un ordinateur équipé d'un Pentium 4 (avec Hyper-Threading) 2.4 GHz avec 512 Mo de RAM.

Une autre mesure de la performance de l'algorithme est en terme de logvraisemblance. On peut ainsi voir [Fig. 7] que l'écart entre la borne inférieure trouvée par SDP et la borne supérieure (l'optimum, en fait, dans les cas où $K \leq 16$) est relativement faible, mais augmente progressivement avec le nombre d'utilisateurs. Pour les cas peu chargés, ces deux bornes sont pratiquement confondues. quand le nombre d'utilisateurs augmente, l'écart entre les deux bornes augmente progressivement, mais reste faible (écart de 4% pour $K = 24$ et $N = 32$ pour VNS). Cet écart reste bon si l'on augmente la capacité et le nombre d'utilisa-

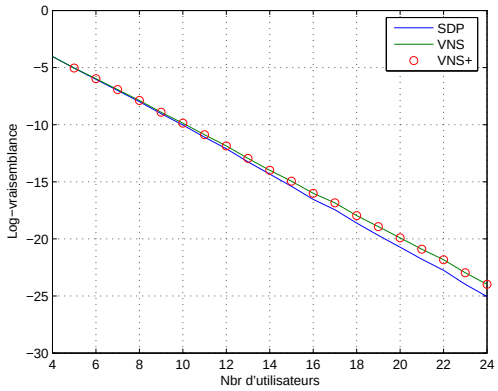

Fig. 7. Écart entre la borne supérieure (VNS) et la borne inférieure (SDP) dans la recherche du maximum de vraisemblance ($N = 32$)

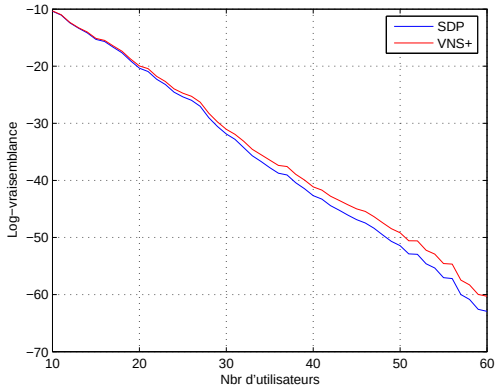

Fig. 8. Écart entre la borne supérieure (VNS) et la borne inférieure (SDP) dans la recherche du maximum de vraisemblance ($N = 64$)

teurs : 3.3% pour le VNS+ lorsque $K = 60$ et $N = 64$ [Fig. 8] ou $N = 96$ [Fig. 9].

6 Conclusion

Les travaux effectués jusqu'à ce jour dans le domaine de la détection multi-utilisateurs reposant sur la programmation semidéfinie sont essentiellement ceux faits par Tan et Rasmussen [10]. Leur méthode de points intérieurs atteint ses

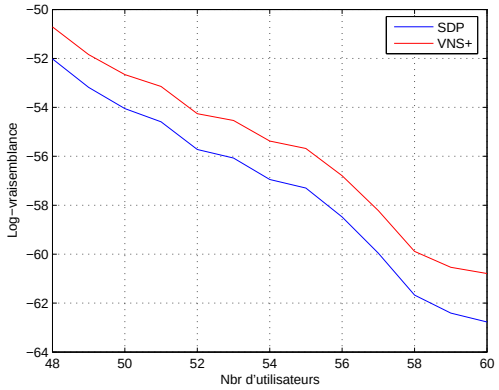

Fig. 9. Écart entre la borne supérieure (VNS) et la borne inférieure (SDP) dans la recherche du maximum de vraisemblance ($N = 96$)

limites lorsque le nombre d'utilisateurs est grand, limite que notre approche permet de dépasser. En effet, il est tout à fait possible, comme nous l'étudions, de travailler sur des instances où 60 utilisateurs simultanés sont présents, et on peut raisonnablement envisager de l'appliquer pour des instances plus grandes. Nos résultats, bien qu'émanant du domaine spécifique de la téléphonie mobile, n'y sont pas limités. Du fait de l'équivalence du problème étudié avec le problème de MAX CUT, nos résultats (et algorithmes) peuvent se transposer vers d'autres domaines reposant également sur le MAX CUT, et réciproquement. Une question reste cependant ouverte : nos expérimentations nous ont montré que tant l'arrondi simple que l'approximation de Goemans et Williamson donnent une vraisemblance proche, mais ce dernier donne des résultats sensiblement moins bons en terme de BER, ce qui le rend, pour cette utilisation, inexploitable. Nous n'avons pas encore déterminé la source de cet effet.

Notations utilisées

v, noté en minuscule, est une variable.
C, noté en majuscule, est une constante.
$\mathbf{v}$, en gras minuscule, est un vecteur.
$\mathbf{M}$, en gras majuscule, est une matrice.
$\mathbf{v}^T$ (resp. $\mathbf{M}^T$) est la transposée de $\mathbf{v}$ (resp. de $\mathbf{M}$).
$\mathbf{v}^H$ (resp. $\mathbf{M}^H$) est la transposée conjuguée de $\mathbf{v}$ (resp. de $\mathbf{M}$), pour les vecteurs (resp. matrices) complexes.
$\mathbb{R}$ (resp. $\mathbb{C}$) est l'ensemble des nombres réels (resp. complexes).
$\mathcal{D}$ est l'ensemble des données (les bits ou symboles) transmises possibles.
$+, -, /, \times$ sont les opérations usuelles sur les nombres, vecteurs ou matrices.

$*$ est le produit de convolution pour deux fonctions, c'est-à-dire la multiplication des transformées de Fourier : $f * g = TF^{-1}(TF(f) \times TF(g))$, avec TF la transformée de Fourier et TF^{-1} la transformée de Fourier inverse.

Références

1. U. Feige and M. X. Goemans. Approximating the value of two power proof systems, with applications to max 2sat and max dicut. In *ISTCS '95 : Proceedings of the 3rd Israel Symposium on the Theory of Computing Systems (ISTCS'95)*, page 182, Washington, DC, USA, 1995. IEEE Computer Society.
2. M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6) :1115–1145, 1995.
3. P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-heuristics, Advances and trends in local search paradigms for optimization*, pages 433–458. Kluwer Academic Publishers, 1999.
4. P. Hansen and N. Mladenović. Developments of variable neighborhood search. *Les Cahiers du GERAD*, G-2001-24, 2001.
5. P. Hansen and N. Mladenović. Variable neighbourhood search. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, chapter 6. Kluwer, 2003.
6. C. Helmberg and F. Rendl. Solving quadratic (0,1)- problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82 :291–315, 1998.
7. M. Laurent and S. Poljak. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra and its Applications*, 223/224(1–3) :439–461, 1995.
8. N. Mladenović and P. Hansen. Variable neighborhood search. *Comput. Oper. Res.*, 24(11) :1097–1100, 1997.
9. J. G. Proakis. *Digital Communications.* McGraw-Hill, New York, 4th edition, 2001.
10. P. H. Tan and L. K. Rasmussen. The application of semidefinite programming for detection in CDMA. *IEEE Journal on Selected Areas in Communications*, 19(8) :1442–1449, 2001.
11. H. L. V. Trees. *Detection, Estimation, and Modulation Theory : Part I.* Wiley, New York, 1968.
12. S. Verdu. *Multiuser Detection.* Cambridge University Press, New York, NY, USA, 1998.
13. H. Wolkowicz. Semidefinite and cone programming bibliography/comments.
14. H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming : Theory, Algorithms, and Applications*, volume 27 of *International series in operations research & management science*. Kluwer Academic Publishers, Dordrecht, The Netherlands / Boston, MA, 2000.
15. J. M. Wozencraft and I. Jacobs. *Principles of Communications Engineering.* Wiley, New York, 1967.
16. U. Zwick. Outward rotations : A tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *STOC*, pages 679–687, 1999.