EPFL - Information, Calcul et Communication - Partie théorique

Série 12: Solutions

1 Distance minimale d'un code binaire

a) On peut vérifier ici que si 3 bits sont effacés, quels qu'ils soient, alors il sera toujours possible d'identifier lequel des 4 messages a été envoyé. Quelques exemples:

- si on reçoit 111??? ou 1?1?0?, alors on sait que le mode de code 111100 a été envoyé;

- si on reçoit ??00?1 ou 11??1?, alors on on sait que le mode de code 110011 a été envoyé.

Par contre, 4 effacements *peuvent* mener à une confusion entre 2 mots de code: par exemple, si on reçoit 11???? (notez cependant que ce n'est pas toujours le cas, car si on reçoit 1????1, alors on sait dans ce cas que le mode de code 110011 a été envoyé).

b) Ce code est garanti de corriger une erreur. En appliquant la règle de la majorité vue au cours, on voit en effet par exemple que si on reçoit 110001, alors seul le mot de code 110000 diffère en 1 bit du mot reçu

Par contre, avec deux erreurs, il *peut* arriver qu'on soit malchanceux et qu'on reçoive par exemple 110000, auquel cas on ne saura pas si le mot de code envoyé était 111100 ou 110011.

c) La distance minimale d du code est égale à 4. En fait, on a une situation un peu particulière ici, car $d(\mathbf{c}_i, \mathbf{c}_j) = 4$ pour *toute* paire de mots de code $(\mathbf{c}_i, \mathbf{c}_j)$.

d) En généralisant les arguments ci-dessus, on voit qu'avec un code avec distance minimale d, il sera toujours possible de corriger jusqu'à $d-1$ effacements et $\lfloor \frac{d-1}{2} \rfloor$ erreurs en utilisant la règle de la majorité.

2 Code de Hamming

a) Pour la première partie, on remarque que:

- si seul $x_1 \neq y_1$, alors $x_5 \neq y_5$, $x_6 \neq y_6$ et $x_7 \neq y_7$
- si seul $x_2 \neq y_2$, alors $x_5 \neq y_5$ et $x_6 \neq y_6$
- si seul $x_3 \neq y_3$, alors $x_5 \neq y_5$ et $x_7 \neq y_7$
- si seul $x_4 \neq y_4$, alors $x_6 \neq y_6$ et $x_7 \neq y_7$

Pour la seconde partie, on observe de même que si (exactement) 2 bits diffèrent entre x_1, x_2, x_3, x_4 et y_1, y_2, y_3, y_4, il n'y pas moyen d'obtenir que les séquences x_5, x_6, x_7 et y_5, y_6, y_7 soient égales.

b) Au vu de ce qui a été dit en a), on a donc la situation suivante:

- si x_1, x_2, x_3, x_4 et y_1, y_2, y_3, y_4 diffèrent en 1 bit, alors la distance entre les 2 mots de code résultants est plus grande ou égale à $1 + 2 = 3$.

- si x_1, x_2, x_3, x_4 et y_1, y_2, y_3, y_4 diffèrent en 2 bits, alors la distance entre les 2 mots de code résultants est plus grande ou égale à $2 + 1 = 3$.

- si x_1, x_2, x_3, x_4 et y_1, y_2, y_3, y_4 diffèrent en 3 ou 4 bits, alors nécessairement, la distance entre les 2 mots de code résultants est plus grande ou égale à 3.

- et finalement, x_1, x_2, x_3, x_4 et y_1, y_2, y_3, y_4 ne diffèrent en aucun bit, alors les 2 mots de code résultants sont les mêmes.

Donc la distance minimale d du code de Hamming est plus grande ou égale à 3.

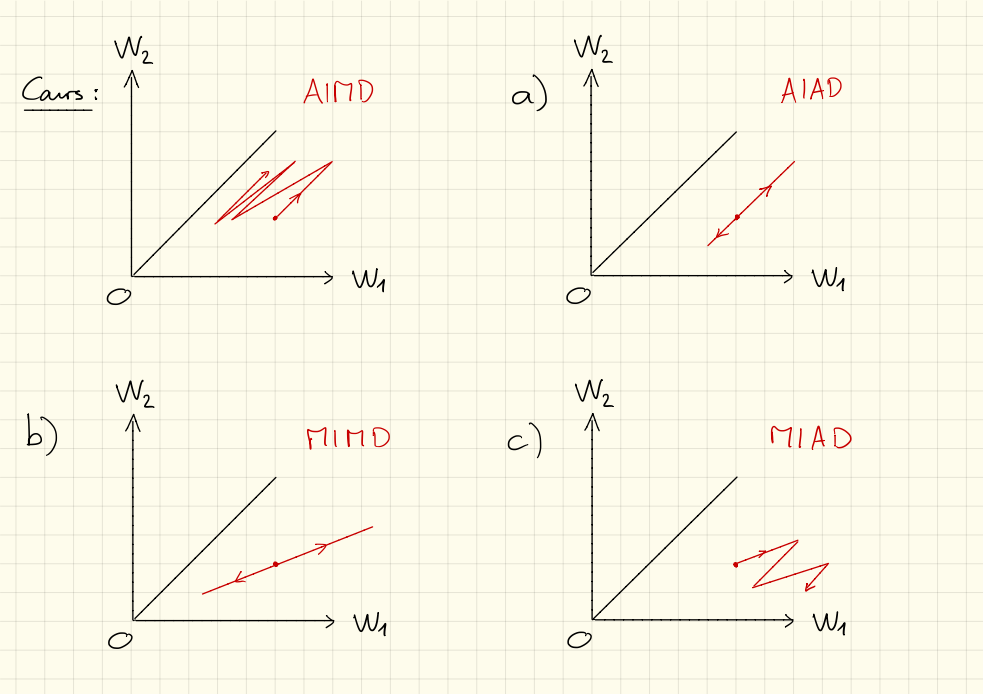
c) La distance entre les mots de code 0000000 et 0100110 vaut 3, donc la distance minimale du code de Hamming vaut exactement 3.

d) Au vu de ce qui a été dit au point d) de l'exercice 1, le code de Hamming corrige jusqu'à $d-1=2$ effacements et $\lfloor \frac{d-1}{2} \rfloor = 1$ erreur.

e) On pourrait être tenté a priori de préférer le code de l'exercice 1, car il corrige plus d'effacements que le code de Hamming, mais il faut également réaliser que le code de l'exercice 1 ne permet que de transmettre un message parmi 4 possibles (les points cardinaux, ou 2 bits en général), tandis que le code de Hamming permet de transmettre 4 bits, donc un message parmi 16 possibles. Si on désire transférer un message parmi 16 possibles avec le code de l'exercice 1, on devra utiliser celui-ci 2 fois de suite, et on utilisera en tout $2 \times 6 = 12$ bits, tandis que le code de Hamming n'utilise que 7 bits en tout pour transmettre ce même message. Ce compromis à respecter entre le nombre de messages qu'il est possible d'envoyer lors d'une transmission et la redondance qu'il est nécessaire d'ajouter à un message afin de corriger plus d'erreurs est une question clé au coeur de la théorie du codage.

3 Algorithme AIMD et autres

Voici l'effet de ces différents algorithmes illustrés sur les graphiques suivants:



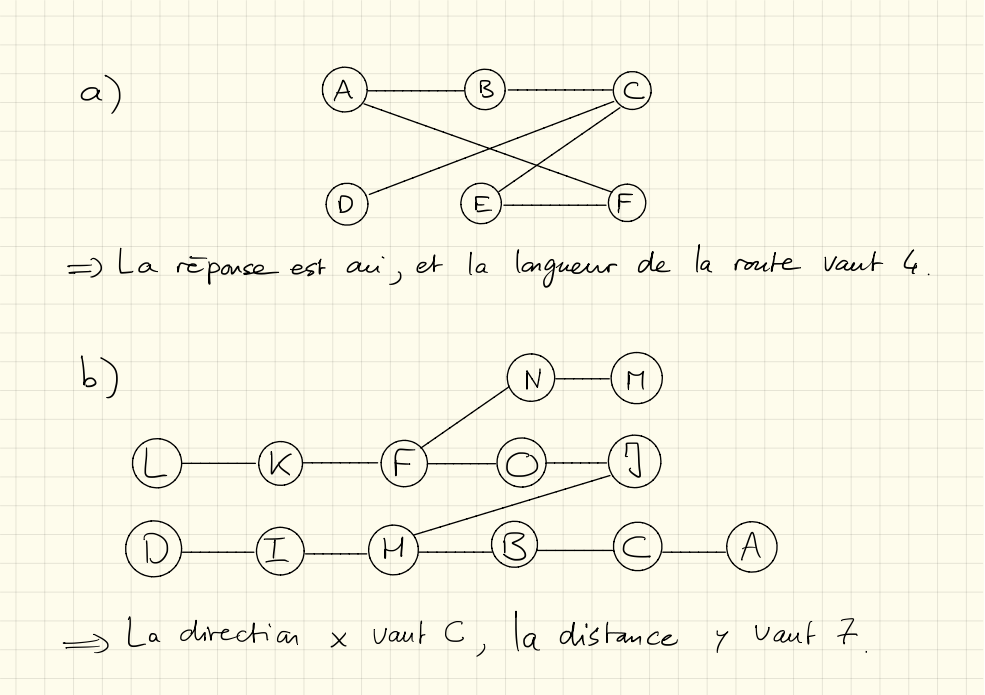
Sur le premier graphique en haut à gauche (cf. cours), on voit l'effet positif de l'algorithme AIMD qui a tendance à faire se rapprocher les valeurs de W_1 et W_2 au cours du temps.

Sur le graphique en haut à droite (algorithme AIAD - partie a), on voit que malgré le fait que W_1 et W_2 varient au cours du temps, les valeurs possibles restent toujours cantonnées sur une droite orientée à 45 degrés qui reste loin de la diagonale $W_1 = W_2$.

Sur le graphique en bas à gauche (algorithme MIMD - partie b), on voit également que malgré le fait que W_1 et W_2 varient au cours du temps, les valeurs possibles restent toujours cantonnées sur une droite partant de 0 et de pente W_2/W_1, qui reste loin de la diagonale $W_1 = W_2$.

Enfin, sur le graphique en bas à droite (algorithme MIAD - partie c), on voit que la situation est encore pire ici, car la paire (W_1, W_2) *s'éloigne* de la diagonale $W_1 = W_2$ au cours du temps, favorisant de plus en plus un utilisateur par rapport à l'autre (en l'occurence ici, l'utilisateur 1, qui avait déjà un léger avantage au départ).

4 Routage



5 Pour le plaisir: codes-barres*

a) Même si entrelacer des barres noirs et blanches rend le code-barres moins lisible, ça permet surtout d'économiser de la place (et donc d'inclure plus de chiffres pour un code-barres de même longueur (en cm)), en utilisant tous les espaces blancs qui seraient inutiles autrement.

b) La distance minimale du code "2 parmi 5" vaut $d = 2$: tous les mots de code sont à distance plus grande ou égale à 2 l'un de l'autre, et en particulier, les mots de code 10001 et 11000 sont exactement à distance 2. Ce code ne peut donc corriger qu'un seul effacement, et aucune erreur. Par contre, il peut *détecter* une erreur: une inversion $0 \longleftrightarrow 1$ va en effet produire une séquence de 5 bits avec 3 bits valant 1 ou au contraire 1 seul bit valant 1, ne faisant clairement pas partie de l'ensemble des mots de code.

c) Si de plus les erreurs sont toutes du même type (p.ex., des inversions du type $0 \to 1$), alors ce code pourra détecter un nombre arbitraire d'erreurs.

d) Dans un code-barres bidimensionnel, on peut a priori placer beaucoup plus d'informations que dans un code-barres unidimensionnel. Dans les exemples donnés, le code-barres unidimensionnel contient 8 chiffres, donc approximativement $8 \times 4 = 32$ bits d'information, tandis que le code-barres bidimensionnel contient l'adresse web `http://www.wikipedia.org/` en code ASCII, donc environ $25 \times 8 = 200$ bits d'information (ou plus).