UNIVERSITÉ LAVAL

Algorithme de Pacman

une méthode efficace pour la construction des codes équilibrés

Mounir Mechqrane

Laboratoire de codification et de théorie d'information de l'université Laval (CODE TI)

Table des matières

Introduction

C’est quoi un bloc de bits équilibré?

Définition

- Un bloc de bits équilibré contient un nombre de bits 0 égal au nombre de bits 1.
- $\{0,1\}_n$ est l’ensemble des blocs binaires de taille n
- $\mathcal{B}_n$ est l’ensemble des blocs équilibrés de taille n.

C’est quoi un bloc de bits équilibré?

Définition

- Un bloc de bits équilibré contient un nombre de bits 0 égal au nombre de bits 1.
- $\{0,1\}_n$ est l’ensemble des blocs binaires de taille n
- $\mathcal{B}_n$ est l’ensemble des blocs équilibrés de taille n.

Note

- $\mathcal{B}_n \subset \{0,1\}_n$
- $|\mathcal{B}_n| = \binom{n}{n/2}$ et $|\{0,1\}_n| = 2^n$, où $\binom{n}{n/2} < 2^n$

C’est quoi un bloc de bits équilibré?

Définition

- Un bloc de bits équilibré contient un nombre de bits 0 égal au nombre de bits 1.
- $\{0,1\}_n$ est l’ensemble des blocs binaires de taille n
- $\mathcal{B}_n$ est l’ensemble des blocs équilibrés de taille n.

Note

- $\mathcal{B}_n \subset \{0,1\}_n$
- $|\mathcal{B}_n| = \binom{n}{n/2}$ et $|\{0,1\}_n| = 2^n$, où $\binom{n}{n/2} < 2^n$

Example

Pour n=2 on a :

$$\{0,1\}_2 = \{00,01,10,11\} \text{ et } \mathcal{B}_2 = \{01,10\}.$$

C’est quoi un code binaire équilibré?

Définitions

- Un dictionnaire qui contient les blocs de bits utilisés dans le codage. Ces blocs sont appelés des mots de code.

C’est quoi un code binaire équilibré?

Définitions

- Un dictionnaire qui contient les blocs de bits utilisés dans le codage. Ces blocs sont appelés des mots de code.
- Ses mots de code sont tous équilibrés.

C’est quoi un code binaire équilibré?

Définitions

- Un dictionnaire qui contient les blocs de bits utilisés dans le codage. Ces blocs sont appelés des mots de code.
- Ses mots de code sont tous équilibrés.

Exemple

Le dictionnaire suivant est un code équilibré de taille 4 :

$$E = \{0011, 0101, 0110, 1001, 1010, 1100\}$$

Pourquoi les codes binaires équilibrés?

W Chaîne d'entrée

Pourquoi les codes binaires équilibrés?

W — Chaîne d'entrée

$w_1 \cdot w_2 \cdot w_3 \cdot \ldots \cdot w_{n-2} \cdot w_{n-1} \cdot w_n$ où $w_{i \in \{1,\ldots,n\}} \in D$ et D définie sur A

Pourquoi les codes binaires équilibrés?

W	Chaîne d'entrée
$w_1 \cdot w_2 \cdot w_3 \cdot \ldots \cdot w_{n-2} \cdot w_{n-1} \cdot w_n$	où $w_{i\in\{1,\ldots,n\}} \in D$ et D définie sur A
$c\downarrow \quad c\downarrow \quad c\downarrow \quad \ldots \quad c\downarrow \quad c\downarrow \quad c\downarrow$	fonction de codage

Pourquoi les codes binaires équilibrés?

						W								Chaîne d'entrée
w_1	$\cdot$	w_2	$\cdot$	w_3	$\cdot$	...	$\cdot$	w_{n-2}	$\cdot$	w_{n-1}	$\cdot$	w_n		où $w_{i\in\{1,\ldots,n\}} \in D$ et D définie sur A
$c\downarrow$		$c\downarrow$		$c\downarrow$		...		$c\downarrow$		$c\downarrow$		$c\downarrow$		fonction de codage
$c(w_1)$		$c(w_2)$		$c(w_3)$		...		$c(w_{n-2})$		$c(w_{n-1})$		$c(w_n)$		où $c(w_{i\in\{1,\ldots,n\}}) \in E$ et E définie sur B

Pourquoi les codes binaires équilibrés?

W	Chaîne d'entrée
$w_1 \cdot w_2 \cdot w_3 \cdot \ldots \cdot w_{n-2} \cdot w_{n-1} \cdot w_n$	où $w_{i\in\{1,\ldots,n\}} \in D$ et D définie sur A
$c\downarrow \quad c\downarrow \quad c\downarrow \quad \ldots \quad c\downarrow \quad c\downarrow \quad c\downarrow$	fonction de codage
$c(w_1) \; c(w_2) \quad c(w_3) \quad \ldots \quad c(w_{n-2}) \quad c(w_{n-1}) \quad c(w_n)$	où $c(w_{i\in\{1,\ldots,n\}}) \in E$ et E définie sur B
$c(w_1) \cdot c(w_2) \cdot c(w_3) \cdot \ldots \cdot c(w_{n-2}) \cdot c(w_{n-1}) \cdot c(w_n)$	Chaîne de sortie

Pourquoi les codes binaires équilibrés?

						W								Chaîne d'entrée
w_1	$\cdot$	w_2	$\cdot$	w_3	$\cdot$	...	$\cdot$	w_{n-2}	$\cdot$	w_{n-1}	$\cdot$	w_n		où $w_{i\in\{1,\ldots,n\}} \in D$ et D définie sur A
$c\downarrow$		$c\downarrow$		$c\downarrow$		...		$c\downarrow$		$c\downarrow$		$c\downarrow$		fonction de codage
$c(w_1)$		$c(w_2)$		$c(w_3)$		...		$c(w_{n-2})$		$c(w_{n-1})$		$c(w_n)$		où $c(w_{i\in\{1,\ldots,n\}}) \in E$ et E définie sur B
$c(w_1)$	$\cdot$	$c(w_2)$	$\cdot$	$c(w_3)$	$\cdot$	...	$\cdot$	$c(w_{n-2})$	$\cdot$	$c(w_{n-1})$	$\cdot$	$c(w_n)$		Chaîne de sortie

Note

La fonction de codage $c : A^* \rightarrow B^*$ doit être injective et définie sur tous les éléments de D.

Pourquoi les codes binaires équilibrés?

W	Chaîne d'entrée
$w_1 \cdot w_2 \cdot w_3 \cdot \ldots \cdot w_{n-2} \cdot w_{n-1} \cdot w_n$	où $w_{i\in\{1,\ldots,n\}} \in D$ et D définie sur A
$c\downarrow \quad c\downarrow \quad c\downarrow \quad \ldots \quad c\downarrow \quad c\downarrow \quad c\downarrow$	fonction de codage
$c(w_1)\ c(w_2)\ c(w_3)\ \ldots\ c(w_{n-2})\ c(w_{n-1})\ c(w_n)$	où $c(w_{i\in\{1,\ldots,n\}}) \in E$ et E définie sur B
$c(w_1) \cdot c(w_2) \cdot c(w_3) \cdot \ldots \cdot c(w_{n-2}) \cdot c(w_{n-1}) \cdot c(w_n)$	Chaîne de sortie

La fonction de codage dépend de la nature des tâches que nous voulons réaliser !

Pourquoi les codes binaires équilibrés?

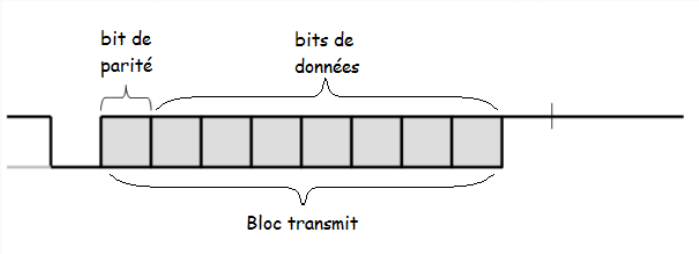
Il y a quatre types de codage :

codage de canal

Pourquoi les codes binaires équilibrés?

Il y a quatre types de codage :

codage de canal



Exemple d'ajout de redondance

Pourquoi les codes binaires équilibrés?

Il y a quatre types de codage :

codage source

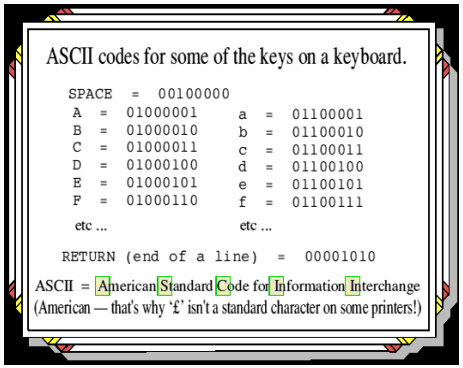
codage de canal

Pourquoi les codes binaires équilibrés?

Il y a quatre types de codage :

codage de canal

codage source



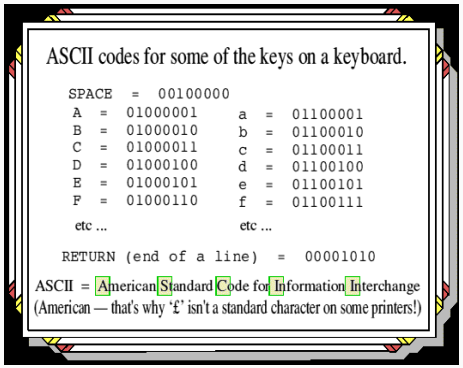
codage de caractères

Pourquoi les codes binaires équilibrés?

Il y a quatre types de codage :

codage de canal

codage source



codage de caractères

codage numérique

Pourquoi les codes binaires équilibrés?

Quatre type de codage, selon les longueurs des mots:

- Codage fixe à variables (FV) : La longueur k des mots dans D est fixe, mais celles des mots dans E sont variables. La longueur k peut être aussi basse que 1.

Pourquoi les codes binaires équilibrés?

Quatre type de codage, selon les longueurs des mots:

- Codage fixe à variables (FV) : La longueur k des mots dans D est fixe, mais celles des mots dans E sont variables. La longueur k peut être aussi basse que 1.
- Codage variable à fixes (VF) : Les mots dans D ont des longueurs variables, mais la longueur l des mots dans E est fixe.

Pourquoi les codes binaires équilibrés?

Quatre type de codage, selon les longueurs des mots:

- Codage fixe à variables (FV) : La longueur k des mots dans D est fixe, mais celles des mots dans E sont variables. La longueur k peut être aussi basse que 1.
- Codage variable à fixes (VF) : Les mots dans D ont des longueurs variables, mais la longueur l des mots dans E est fixe.
- Codages variable à variable (VV) : Les mots dans D et dans E ont des longueurs variables.

Pourquoi les codes binaires équilibrés?

Quatre type de codage, selon les longueurs des mots:

- Codage fixe à variables (FV) : La longueur k des mots dans D est fixe, mais celles des mots dans E sont variables. La longueur k peut être aussi basse que 1.
- Codage variable à fixes (VF) : Les mots dans D ont des longueurs variables, mais la longueur l des mots dans E est fixe.
- Codages variable à variable (VV) : Les mots dans D et dans E ont des longueurs variables.
- Codage fixes à fixes (FF) : La longueur k des mots dans D et la longueur l des mots dans E est fixe.

Pourquoi les codes binaires équilibrés?

Le codage par codes équilibrés est un codage FF détecteur d'erreurs!!

Pourquoi les codes binaires équilibrés?

Le codage par codes équilibrés est un codage FF détecteur d'erreurs!!

Ou tout simplement nous pouvons dire que:
Les codes équilibrés sont des codes FF détecteurs d'erreurs.

Pourquoi les codes binaires équilibrés?

Les codes équilibrés trouvent plusieurs applications dans plusieurs domaines :

Détection des erreurs unidirectionnelles

Pourquoi les codes binaires équilibrés?

Les codes équilibrés trouvent plusieurs applications dans plusieurs domaines :

Détection des erreurs unidirectionnelles

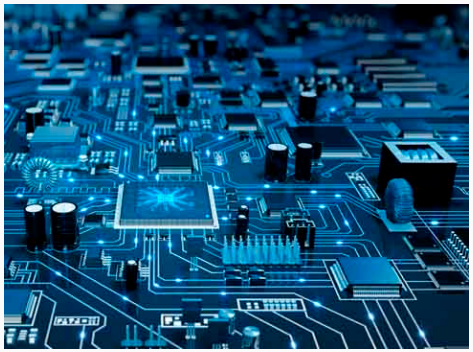
Communications par fibre optique

Pourquoi les codes binaires équilibrés?

Les codes équilibrés trouvent plusieurs applications dans plusieurs domaines :

Détection des erreurs unidirectionnelles

Communications par fibre optique



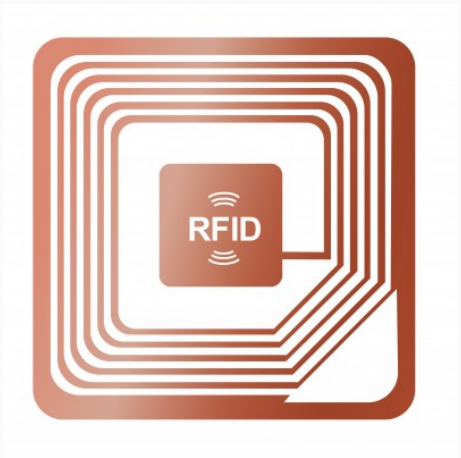
Systèmes VLSI

Pourquoi les codes binaires équilibrés?

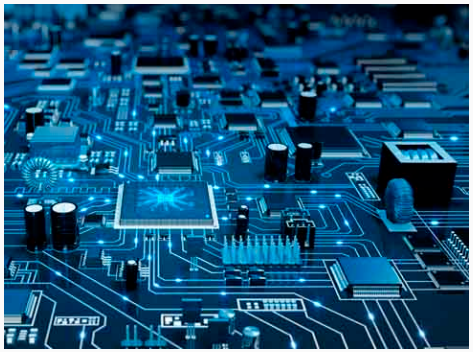
Les codes équilibrés trouvent plusieurs applications dans plusieurs domaines :

Détection des erreurs unidirectionnelles

Communications par fibre optique



Communication RFID



Systèmes VLSI

Problématique

Problématique

- Comment utiliser efficacement les codes binaires équilibrés ?

Détails de la problématique

L'efficacité ici signifie trois choses :

(A) L'efficacité en redondance : Les codes équilibrés contiennent par nature de la redondance (codage de canal). Cependant, cette redondance doit être minimale.

(B) L'efficacité spatiale : utilisation raisonnable des ressources mémoire.

(C) L'efficacité calculatoire : temps de calcul raisonnable.

Objectif

Trouver une manière simple est efficace pour coder et décoder un bloc source de n-bits en bloc équilibré de m-bits.

Autrement dit, nous devons concevoir un système de codage/décodage FF efficace définie comme suit :

$$\begin{cases} Enc : 2^n \rightarrow 2^m \\ Dec : 2^m \rightarrow 2^n \end{cases}$$

(A) L'efficacité en redondance

Note

- Un mot de code équilibré de taille m peut coder $\log_2 \binom{m}{m/2}$ bits source. Autrement dit, pour coder 2^n blocs source de taille n, on a besoin des blocs équilibrés de taille m de façon à ce que $\binom{m}{m/2} \geq 2^n$ [3].

(A) L'efficacité en redondance

Note

- Un mot de code équilibré de taille m peut coder $\log_2 \binom{m}{m/2}$ bits source. Autrement dit, pour coder 2^n blocs source de taille n, on a besoin des blocs équilibrés de taille m de façon à ce que $\binom{m}{m/2} \geq 2^n$ [3].
- Le nombre de bits de parité p pour créer les blocs équilibrés de taille m est donc :

$$p = m - \log_2 \binom{m}{m/2} \tag{1}$$

(A) L'efficacité en redondance

Note

- Un mot de code équilibré de taille m peut coder $\log_2 \binom{m}{m/2}$ bits source. Autrement dit, pour coder 2^n blocs source de taille n, on a besoin des blocs équilibrés de taille m de façon à ce que $\binom{m}{m/2} \geq 2^n$ [3].
- Le nombre de bits de parité p pour créer les blocs équilibrés de taille m est donc :

$$p = m - \log_2 \binom{m}{m/2} \tag{1}$$

- Approximativement, la valeur de p est [3]:

$$p \approx \frac{1}{2} \log_2 m + 0.326, m \gg 1 \tag{2}$$

(A) L'efficacité en redondance

Notation

Soit $\bar{\cdot}$ l'opérateur qui complémente les bits ; c'est-à-dire $\bar{0} = 1$ et $\bar{1} = 0$. Nous étendons cet opérateur pour qu'il fonctionne sur les séquences de bits.

Exemple

1	0	1	0	1	1	0	1

W_n

(A) L'efficacité en redondance

Notation

Soit $\bar{\cdot}$ l'opérateur qui complémente les bits ; c'est-à-dire $\bar{0} = 1$ et $\bar{1} = 0$. Nous étendons cet opérateur pour qu'il fonctionne sur les séquences de bits.

Exemple

1	0	1	0	1	1	0	1

w_n

0	1	0	1	0	0	1	0

$\bar{w}_n$

(A) L'efficacité en redondance

Notation

Soit $\bar{\cdot}$ l'opérateur qui complémente les bits ; c'est-à-dire $\bar{0} = 1$ et $\bar{1} = 0$. Nous étendons cet opérateur pour qu'il fonctionne sur les séquences de bits.

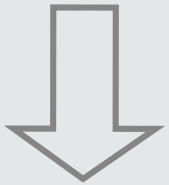
Exemple

1	0	1	0	1	1	0	1

w_n

0	1	0	1	0	0	1	0

$\bar{w}_n$

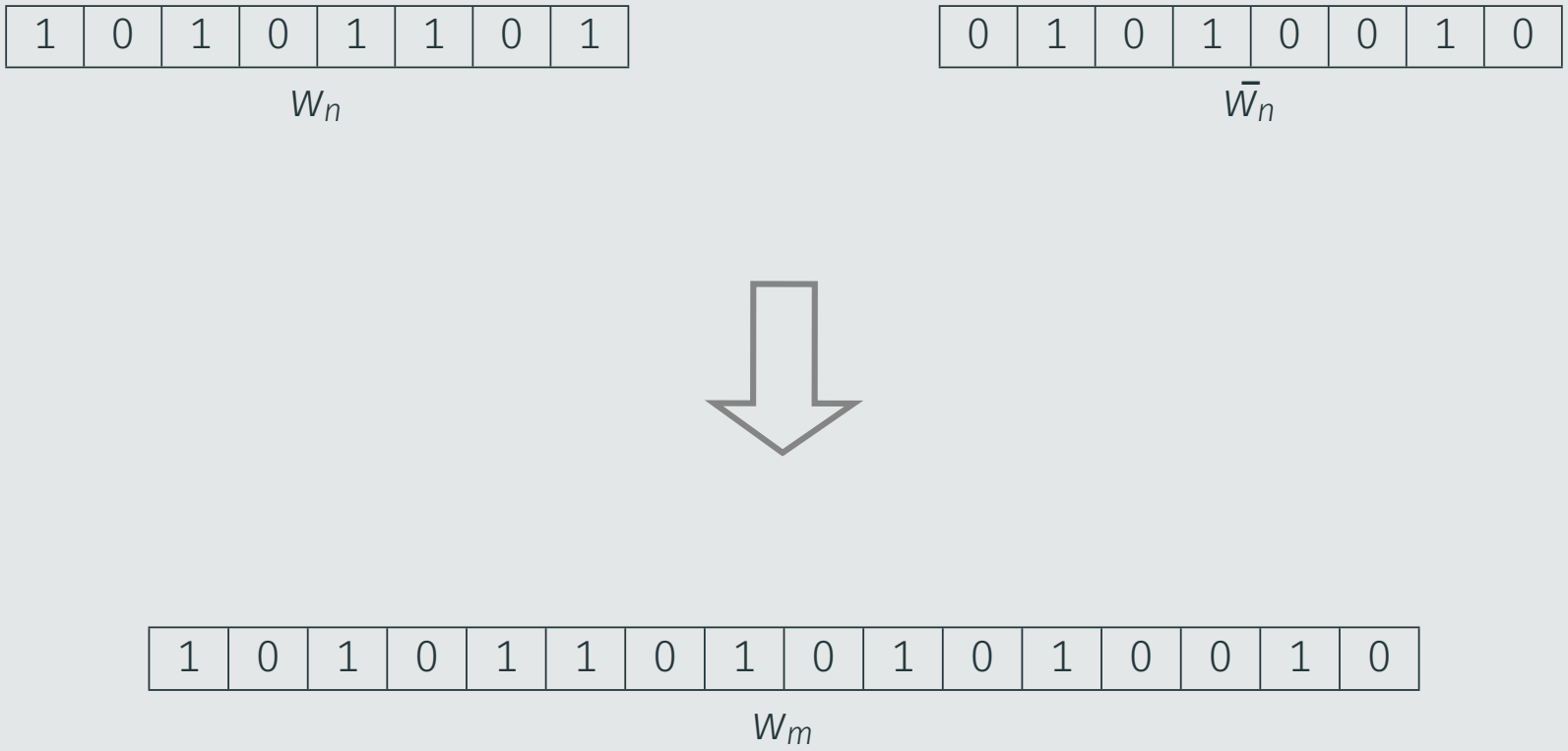


(A) L'efficacité en redondance

Notation

Soit $\bar{\cdot}$ l'opérateur qui complémente les bits ; c'est-à-dire $\bar{0} = 1$ et $\bar{1} = 0$. Nous étendons cet opérateur pour qu'il fonctionne sur les séquences de bits.

Exemple



(A) L'efficacité en redondance

Notation

Soit $\bar{\cdot}$ l'opérateur qui complémente les bits ; c'est-à-dire $\bar{0} = 1$ et $\bar{1} = 0$. Nous étendons cet opérateur pour qu'il fonctionne sur les séquences de bits.

Exemple

1	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0

$$w_m = w_n \cdot \bar{w}_n$$

$$m = 2 \times n.$$

$$\text{Pour } m \gg 1,\ p = m/2 \gg \tfrac{1}{2} \log_2 m + 0.326$$

(B, C) L'efficacité spatiale et calculatoire

Lemme de Sperner [4]

La meilleure façon pour créer des codes équilibrés efficaces est la construction de la liste de tous les mots de codes de longueur $m = n + p$, où p est le nombre de bits de parité (le minimum de bits redondants à ajouter).

En pratique, nous pouvons appliquer le lemme de Sperner en utilisant les tables de consultation (Lookup Tables) :

Exemple

- Pour n = 2 la table de codage est :

Mot d'entrée	Code équilibré
00	0011
01	0101
10	0110
11	1001

(B, C) L'efficacité spatiale et calculatoire

Exemple (Suite)

- Pour n = 4 la table de codage est :

Mot d'entrée	Code équilibré	Mot d'entrée	Code équilibré	Mot d'entrée	Code équilibré	Mot d'entrée	Code équilibré
0000	000111	0100	010011	1000	011010	1100	100110
0001	001011	0101	010101	1001	011100	1101	101001
0010	001101	0110	010110	1010	100011	1110	101010
0011	001110	0111	011001	1011	100101	1111	101100

- Pour n = 6 la table de codage est :

Mot d'entrée	Code équilibré	Mot d'entrée	Code équilibré	Mot d'entrée	Code équilibré	Mot d'entrée	Code équilibré
000000	00001111	000100	00011110	001000	00101110	001100	00111001
000001	00010111	000101	00100111	001001	00110011	001101	00111010
000010	00011011	000110	00101011	001010	00110101	001110	00111100
000011	00011101	000111	00101101	001011	00110110	001111	01000111

⋮ ⋮ ⋮ ⋮

(B, C) L'efficacité spatiale et calculatoire

Exemple (Suite 2)

- Pour n = 512 la table de codage nécessite ...

Explosion Big Bang

$$2^{512} > 2^{326} = 2^{2+4\times 81} = 4 \times 16^{81} > 4 \times 10^{81}$$

Une autre alternative pour appliquer le lemme de Sperner est l'utilisation du codage énumerative [1].

Travaux antérieurs

Algorithme de Knuth

Complémenter les k premiers bits pour équilibrer le mot de bits !!!

Algorithme de Knuth

Il y a c possibilités de k !!![2]

$1 \leq c \leq \frac{n}{2}$

Knuth choisit par défaut la 1^e possibilité

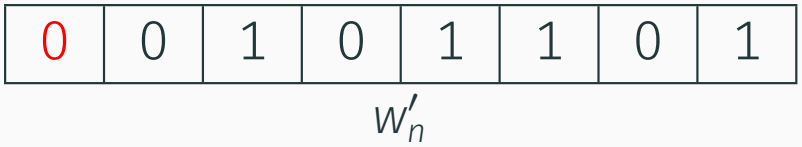
1	0	1	0	1	1	0	1

w_n

Complémenter les k premiers bits pour équilibrer le mot de bits !!!

Algorithme de Knuth

Algorithme de Knuth



Algorithme de Knuth

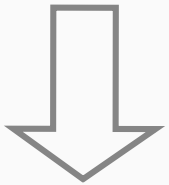
u

w'_n

Algorithme de Knuth

u

w'_n



w_m

La redondance de l'algorithme de Knuth est [3]:

$$p_k \approx \log_2 m + \frac{1}{2} \log_2 \log_2 m, m \gg 1 \tag{3}$$

Algorithme de Knuth

Performances de l'algorithme de Knuth

- La simplicité de l'implementation.
- Utilisation raisonnable des ressources.
- **La liberté de selection non exploitée !!!**
- **Le nombre de bits de parité égale au double du seuil minimal !!!**

Algorithme de Knuth

Remarques

- La liberté de sélection montre qu'il y a une multiplicité d'encodage (ME) dans l'algorithme de Knuth.
- L'algorithme de Knuth n'est pas optimal en redondance à cause de ME.

w_n								k	u							w'_n							
1	0	1	0	1	1	0	1	1	0	0	1	0	1	1	0	0	1	0	1	1	0	1
								3	0	0	1	1	1	0	0	1	0	0	1	1	0	1
								5	0	1	0	1	0	1	0	1	0	1	0	1	0	1
								7	0	1	1	0	0	1	0	1	0	1	0	0	1	1

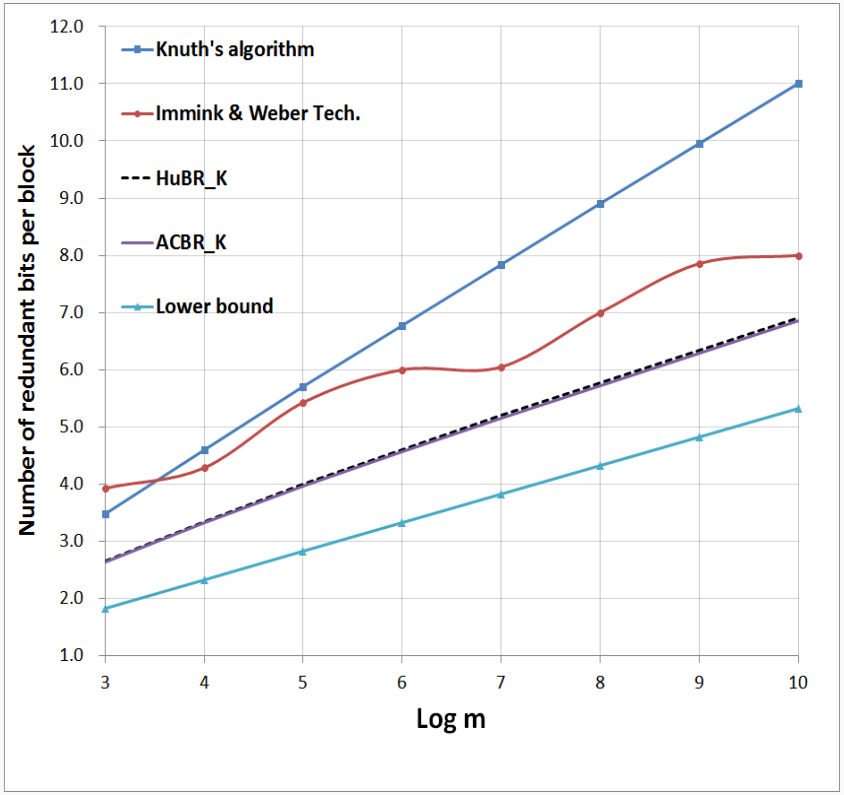
Autres travaux

L'exploitation de ME permet de réduire la redondance de l'algorithme de Knuth par [2] :

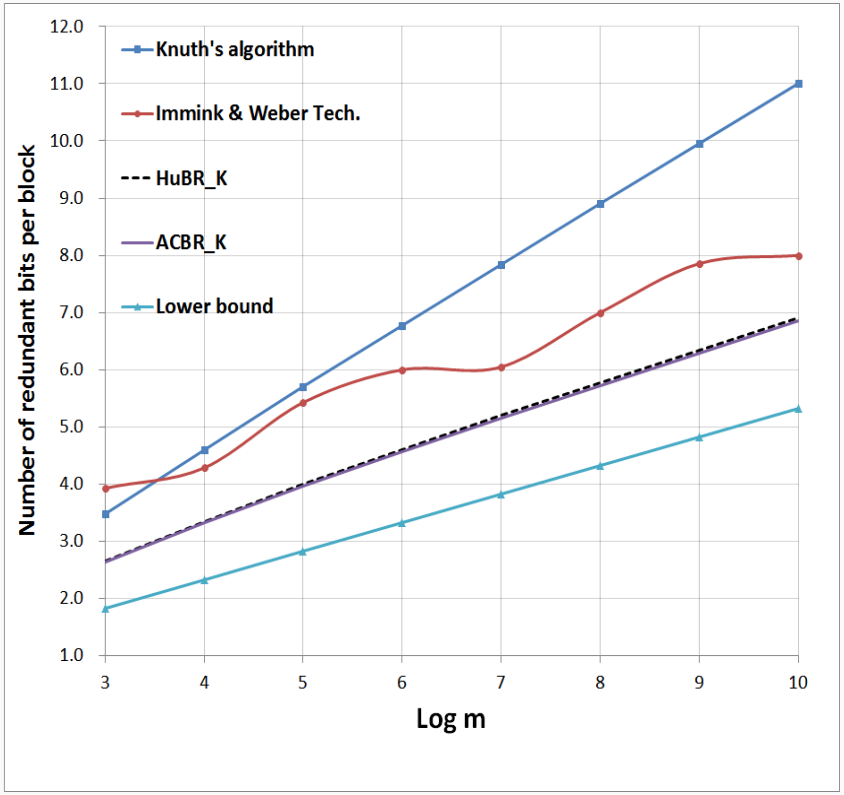
$$A_{SF}(m) \approx \frac{1}{2} \log_2 m - 0.916 \tag{4}$$

Weber & Immink et Al-Rababa'a et al. ont essayé d'exploiter la multiplicité d'encodage pour réduire la redondance de l'algorithme de Knuth. Leurs résultats sont présentés dans l'acétate suivante.

Autres travaux



Autres travaux



Spécification de l'objectif

La conception d'un nouveau algorithme qui élimine l'écart par rapport au seuil minimal de la redondance toute en restant efficace..

Notions préliminaire

Permutation

La notation conventionnelle d'une permutation

- La notation conventionnelle d'une permutation de m éléments est $(a_1, ..., a_m)$, où $a_i \neq a_j$ lorsque $1 \leq i < j \leq m$.

Permutation

La notation conventionnelle d'une permutation

- La notation conventionnelle d'une permutation de m éléments est $(a_1, ..., a_m)$, où $a_i \neq a_j$ lorsque $1 \leq i < j \leq m$.
- Soit S un ensemble de taille m. On dit que $(a_1, ..., a_m)$ est une permutation des éléments de S (ou, par abus de langage, une permutation de S) si $\{a_1, ..., a_m\} = S$.

Permutation

La notation conventionnelle d'une permutation

- La notation conventionnelle d'une permutation de m éléments est $(a_1, ..., a_m)$, où $a_i \neq a_j$ lorsque $1 \leq i < j \leq m$.
- Soit S un ensemble de taille m. On dit que $(a_1, ..., a_m)$ est une permutation des éléments de S (ou, par abus de langage, une permutation de S) si $\{a_1, ..., a_m\} = S$.
- Nous définissons $\mathcal{P}_m$ comme l'ensemble des permutations de $\{1, ..., m\}$, où $m \in 2\mathbb{N}$.

Permutation

La notation conventionnelle d'une permutation

- La notation conventionnelle d'une permutation de m éléments est $(a_1, ..., a_m)$, où $a_i \neq a_j$ lorsque $1 \leq i < j \leq m$.
- Soit S un ensemble de taille m. On dit que $(a_1, ..., a_m)$ est une permutation des éléments de S (ou, par abus de langage, une permutation de S) si $\{a_1, ..., a_m\} = S$.
- Nous définissons $\mathcal{P}_m$ comme l'ensemble des permutations de $\{1, ..., m\}$, où $m \in 2\mathbb{N}$.
- Nous notons la permutation identité de $\mathcal{P}_m$ par π_0.

Permutation

La notation conventionnelle d'une permutation

- La notation conventionnelle d'une permutation de m éléments est $(a_1, ..., a_m)$, où $a_i \neq a_j$ lorsque $1 \leq i < j \leq m$.
- Soit S un ensemble de taille m. On dit que $(a_1, ..., a_m)$ est une permutation des éléments de S (ou, par abus de langage, une permutation de S) si $\{a_1, ..., a_m\} = S$.
- Nous définissons $\mathcal{P}_m$ comme l'ensemble des permutations de $\{1, ..., m\}$, où $m \in 2\mathbb{N}$.
- Nous notons la permutation identité de $\mathcal{P}_m$ par π_0.

Exemple

$\pi = (4, 1, 3, 2)$ est une permutation de $\{1, 2, 3, 4\}$.

Permutation

La notation par index d'une permutation

- En notation indexée, nous représentons une permutation indexée η sous la forme $\langle \iota_m, ..., \iota_1 \rangle$, où $1 \leq \iota_i \leq i$ pour $1 \leq i \leq m$.

Permutation

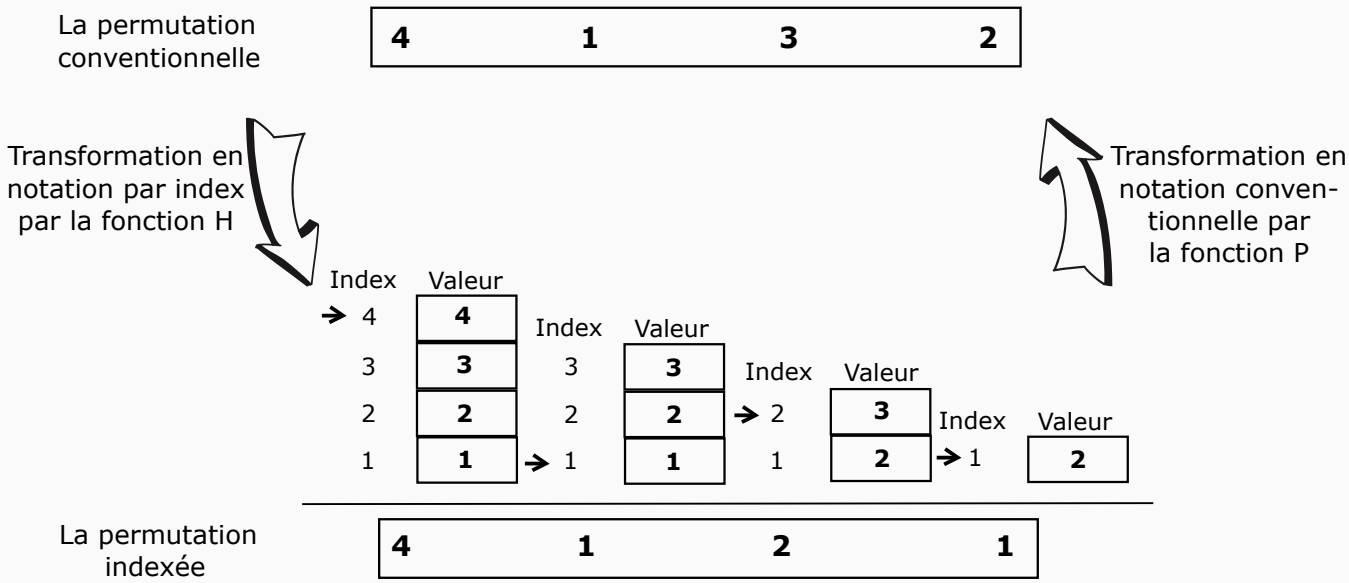
La notation par index d'une permutation

- En notation indexée, nous représentons une permutation indexée η sous la forme $\langle \iota_m, ..., \iota_1 \rangle$, où $1 \leq \iota_i \leq i$ pour $1 \leq i \leq m$.
- Nous définissons $\mathcal{H}_m$ comme l'ensemble des permutations indexées de taille m.

Permutation

La notation par index d'une permutation

- En notation indexée, nous représentons une permutation indexée η sous la forme $\langle \iota_m, ..., \iota_1 \rangle$, où $1 \leq \iota_i \leq i$ pour $1 \leq i \leq m$.
- Nous définissons $\mathcal{H}_m$ comme l'ensemble des permutations indexées de taille m.



Exemple de la notation par index

Permutation

Permutations et bloc équilibrés

- À partir d'une permutation, nous somme capable de produire un code équilibré par l'extraction de la parité.

Permutation

Permutations et bloc équilibrés

- À partir d'une permutation, nous somme capable de produire un code équilibré par l'extraction de la parité.
- Nous utilisons pour cela la fonction *mod*.

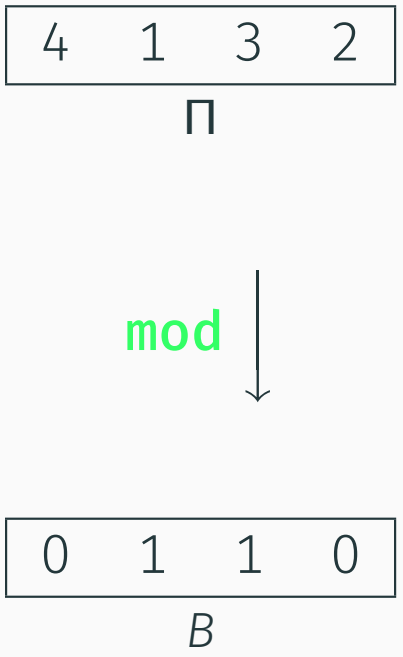
Permutation

Permutations et bloc équilibrés

- À partir d'une permutation, nous somme capable de produire un code équilibré par l'extraction de la parité.
- Nous utilisons pour cela la fonction *mod*.

Note

- Reste à trouver un moyen pour incorporer les bits sources dans Π.



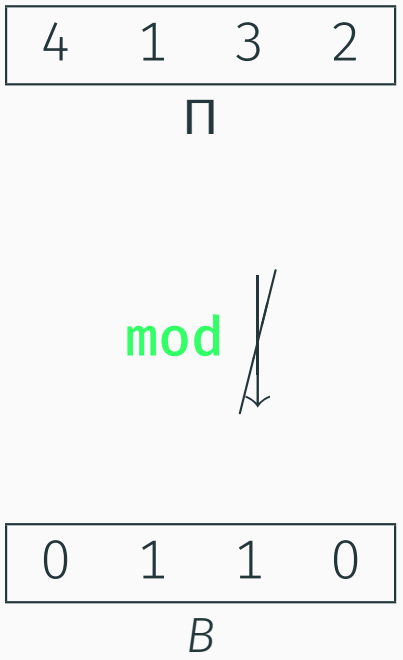
Permutation

Permutations et bloc équilibrés

- À partir d'une permutation, nous somme capable de produire un code équilibré par l'extraction de la parité.
- Nous utilisons pour cela la fonction *mod*.

Note

- Reste à trouver un moyen pour incorporer les bits sources dans Π.
- Un autre problème est comment exploiter Π après l'extraction de B.



Permutation

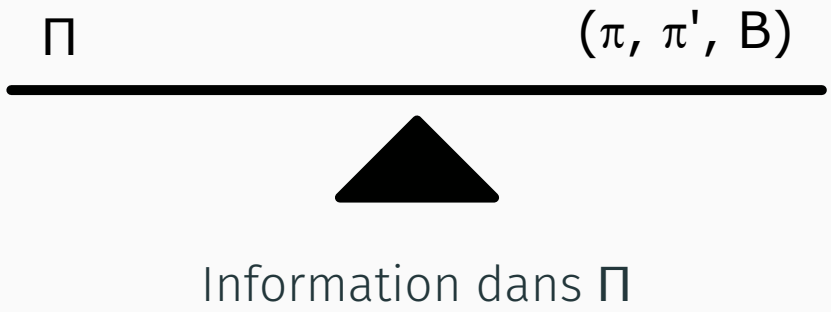
Note (suite)

- En effet, l'extraction de *B* ne consomme pas la totalité de l'information contenue dans Π.

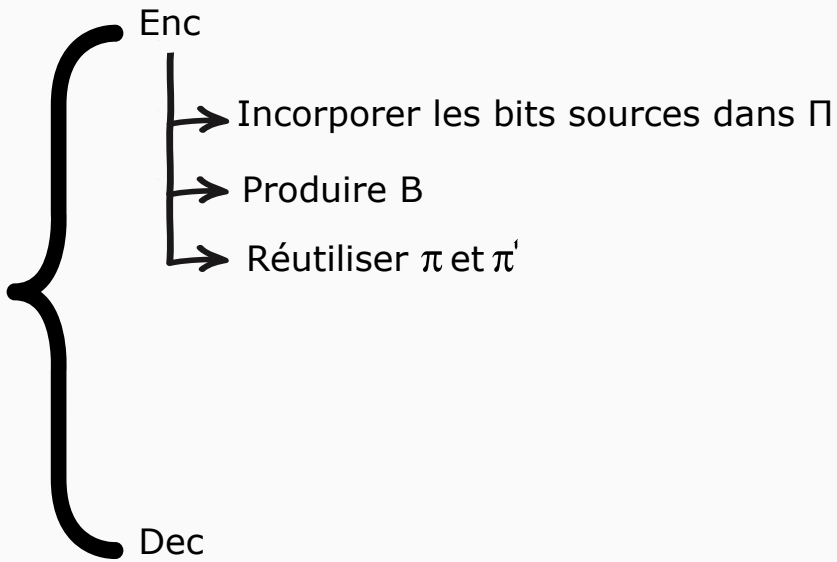
Permutation

Note (suite)

- En effet, l'extraction de B ne consomme pas la totalité de l'information contenue dans Π.
- $\Pi \Leftrightarrow (\pi, \pi', B)$.



Permutation

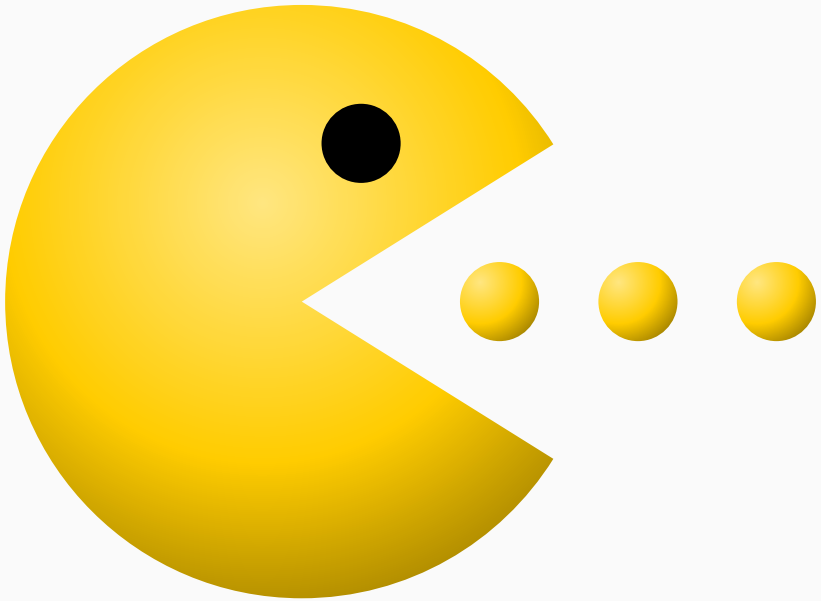


Système de codage/décodage

Réutilisation de π et π'

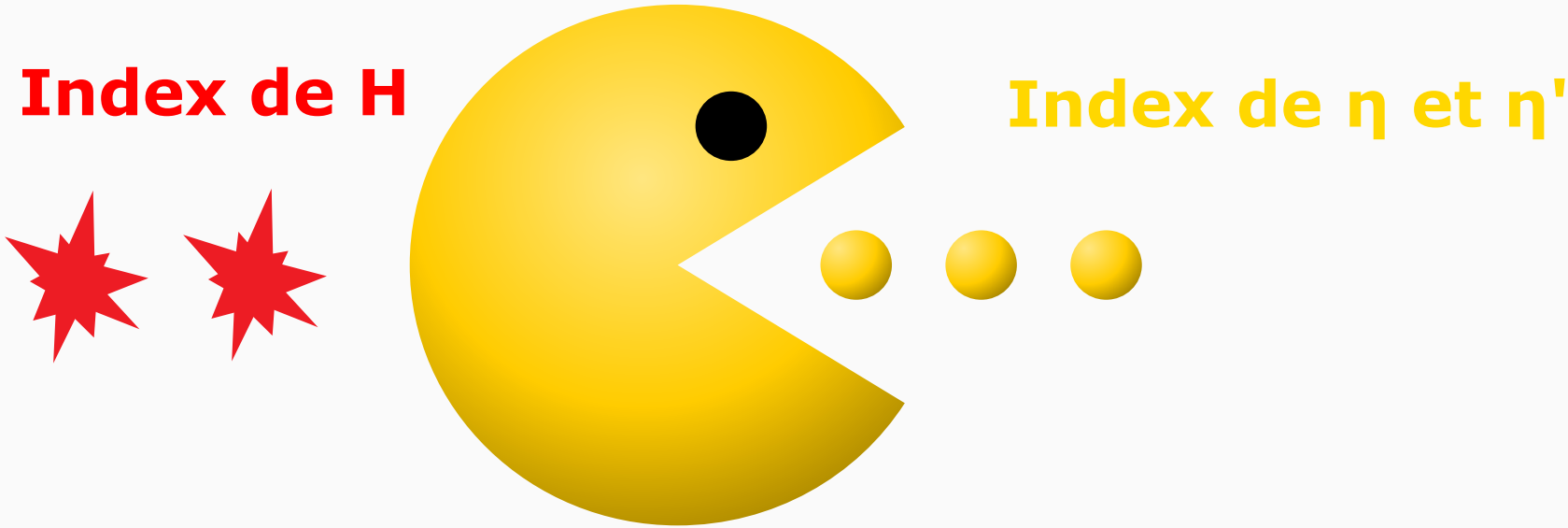
- Transformation en notation par index : $\eta = H(\pi)$ et $\eta' = H(\pi')$
- Transformation de η et η' en H.
- Transformation de H en notation conventionnelle $\Pi = P(H)$.

Pacman



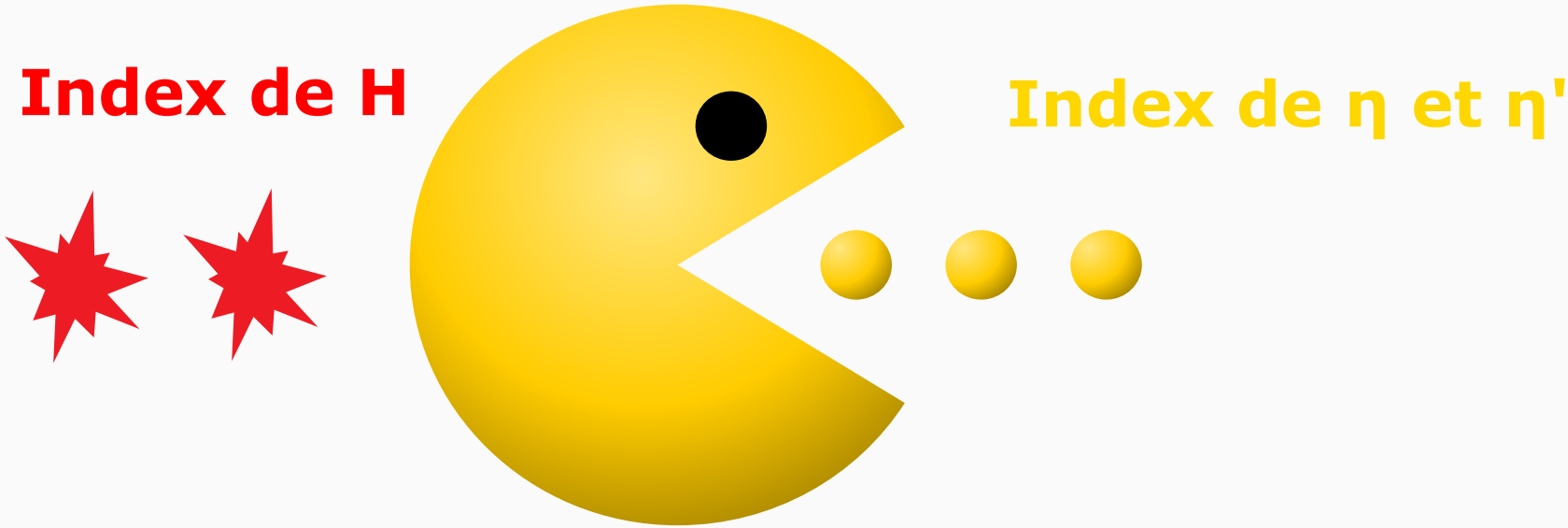
Le PAC-MAN ordinaire

Pacman

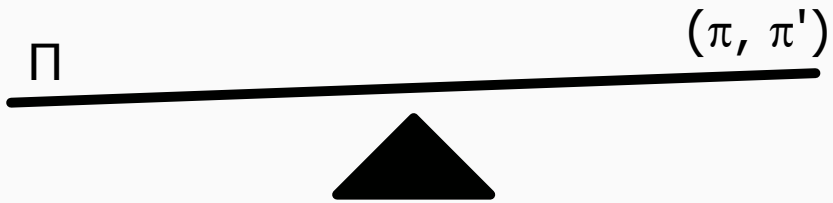


Le PAC-MAN spécial

Pacman

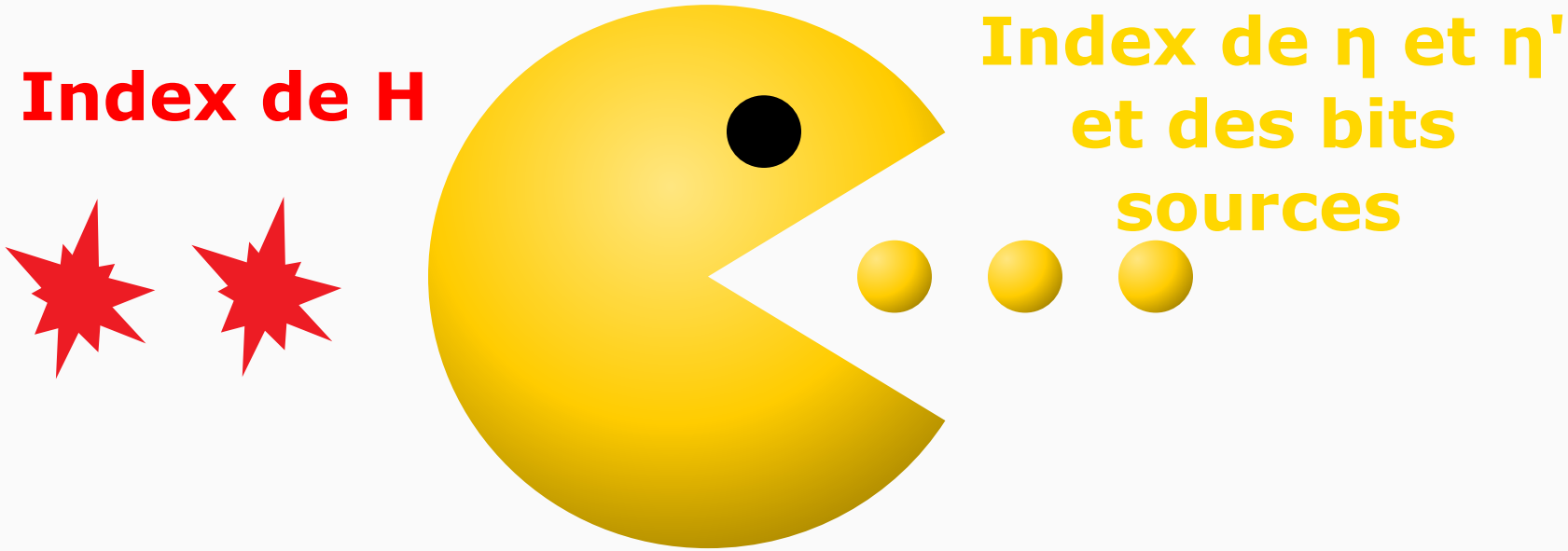


Le PAC-MAN spécial



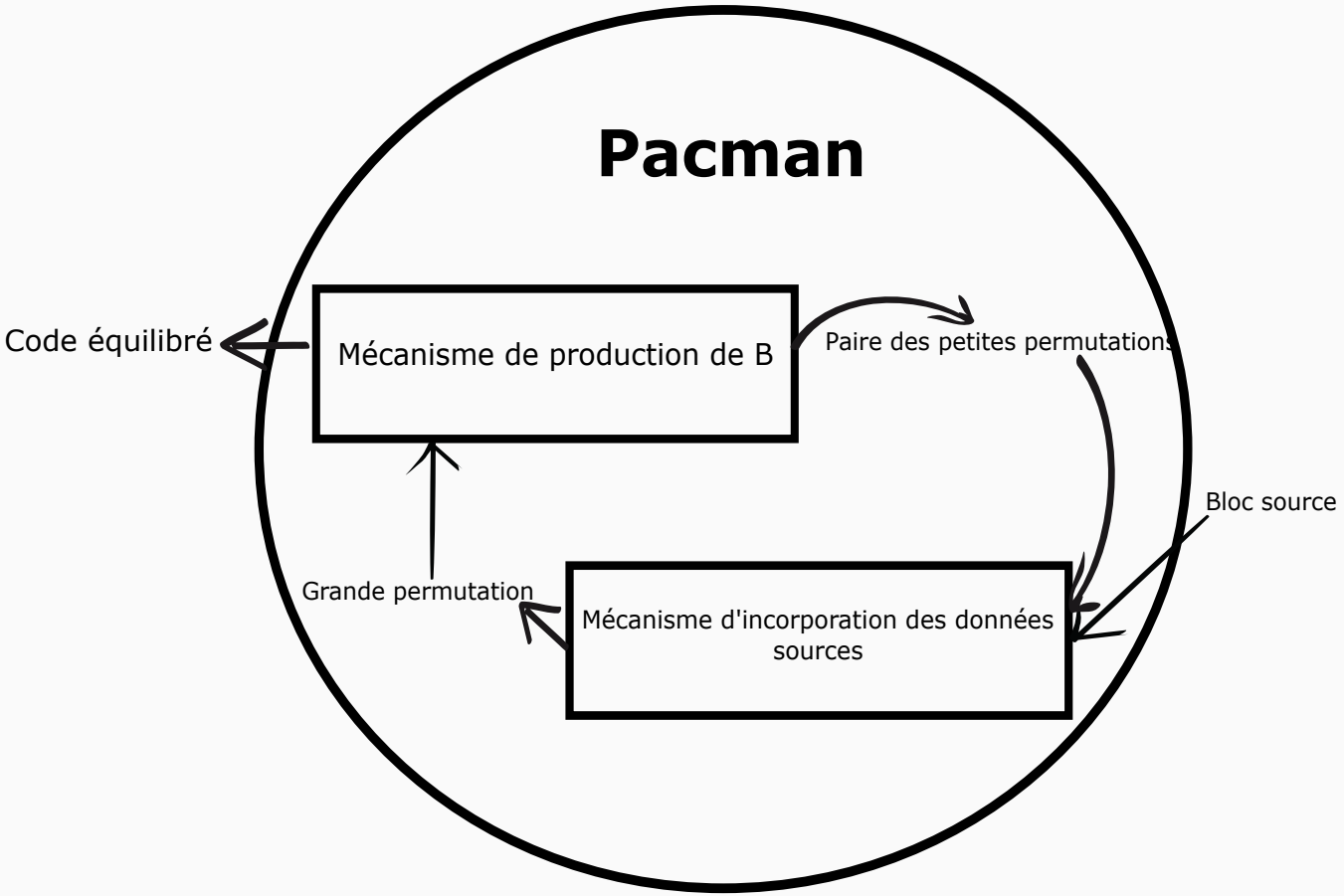
Information dans π et π'

Pacman



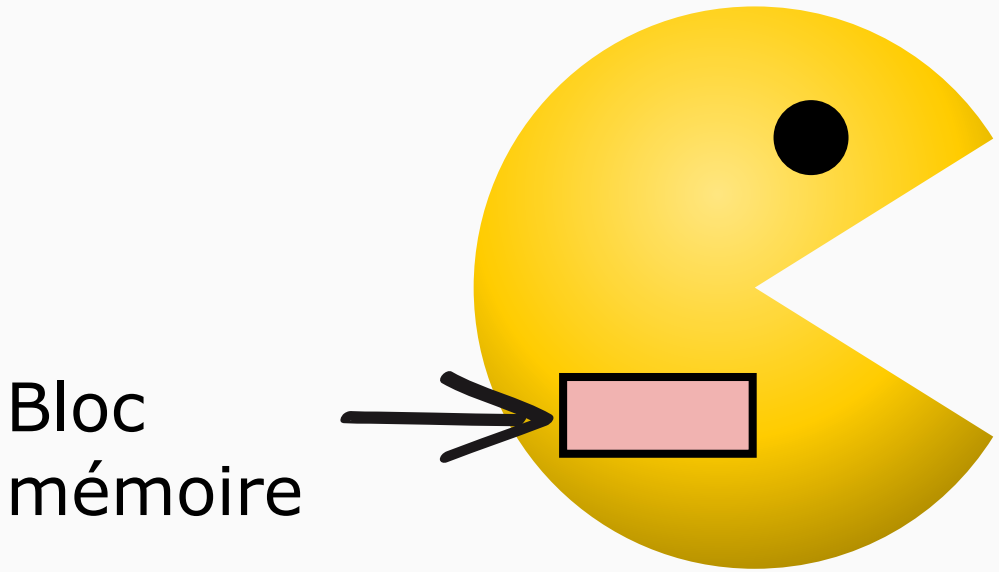
Le PAC-MAN spécial

Pacman



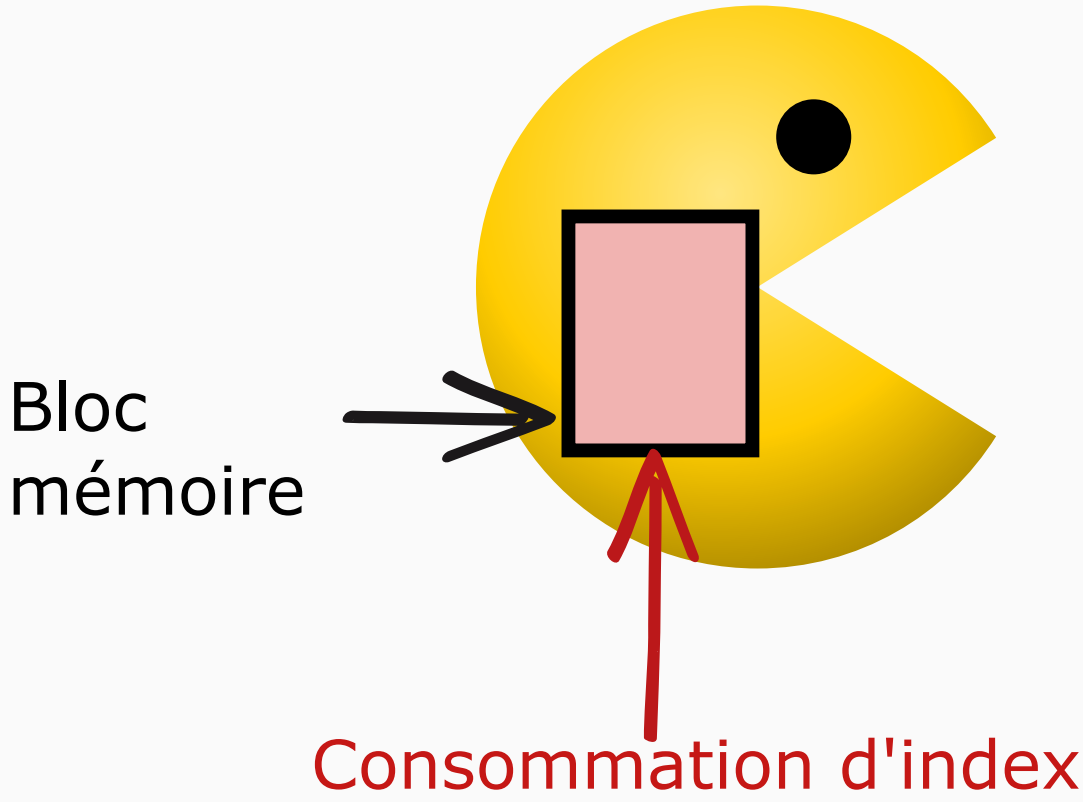
Cycle de Pacman

Pacman



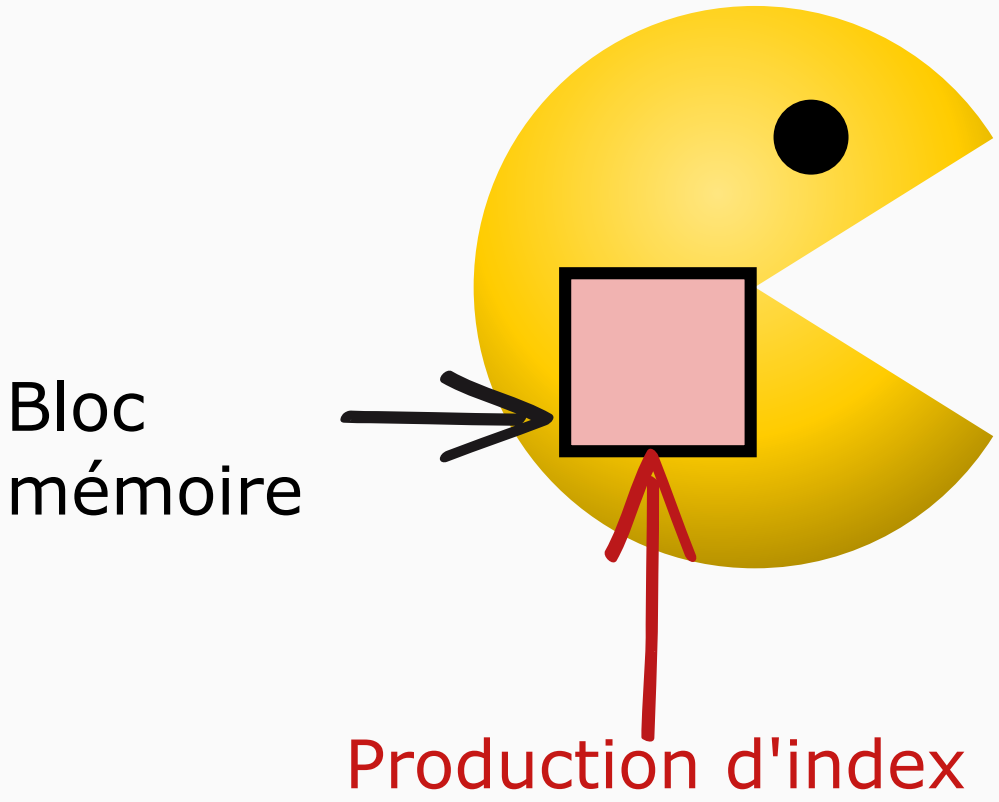
Mémoire de Pacman

Pacman



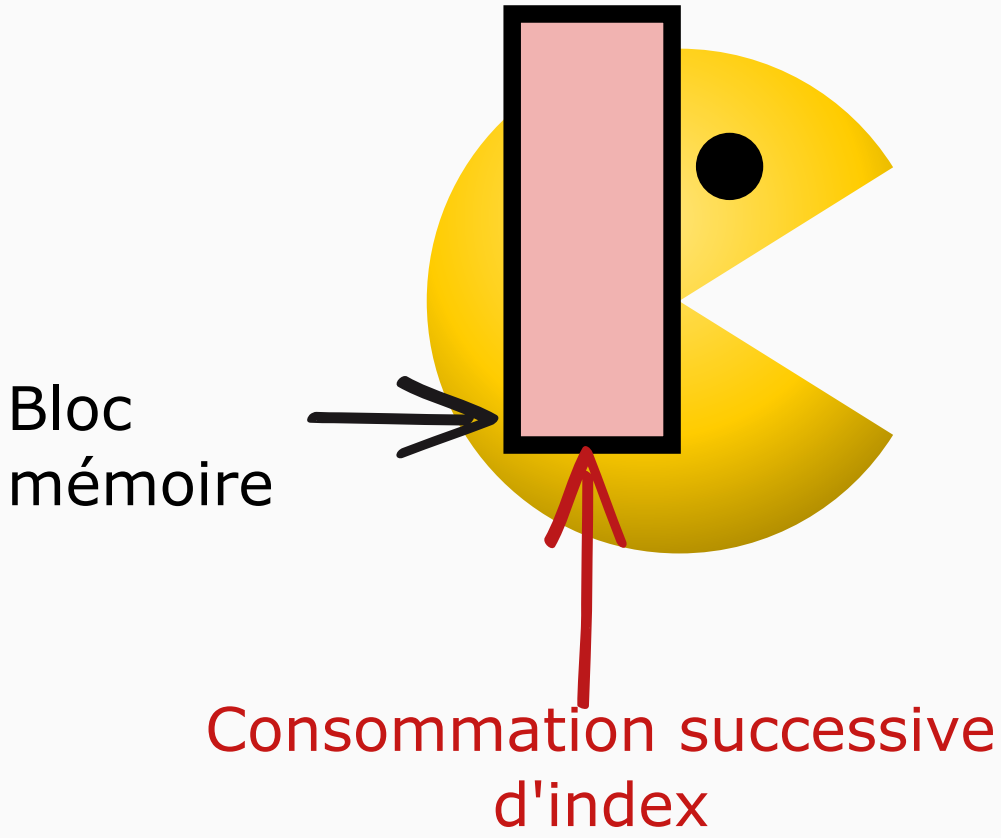
L’effet de la consommation d’index sur la mémoire

Pacman



L’effet de la production d’index sur la mémoire

Pacman



L’effet de la consommation consécutive d’index sur la mémoire

Note

Un contrôle est appliqué sur la taille de la mémoire de Pacman.

Mémoire limitée

Controle de la mémoire

- Nous mesurons la taille de la mémoire de Pacman avant et après chaque opération.
- Nous considérons que Pacman se souvient à chaque fois d'une seule valeur naturelle v dans l'interval $1, ..., \sigma$ où $\sigma \geq 1$.
- Soient v, σ, v' et σ' la valeur et la taille de la mémoire avant et après une opération donnée :
  - L'effet d'une opération de consommation sur la mémoire est :
$$\begin{cases} v' = \rho \times (v - 1) + i \\ \sigma' = \rho \times \sigma \end{cases}$$
  - L'effet d'une opération de production sur la mémoire est :
$$\begin{cases} (i, v') = (((v - 1)\ mod\ \rho) + 1, \lceil v/\rho \rceil) \\ \sigma' = \lceil \sigma/\rho \rceil \end{cases}$$

Mémoire limitée

Remarque

- Les bits sources sont considérés comme des index de taille 2. (Ajustement par 1)
- Le seul ajout de redondance est : $\sigma' = \lceil \sigma/\rho \rceil$. Nous avons dans ce cas $\sigma' \leq \frac{\sigma+\rho-1}{\rho}$. Donc $\lim_{\sigma\to\infty} \frac{\rho-1}{\sigma'} = 0$.

Méthodologie

Programmation de l'algorithme de Pacman

Programmation de Pacman

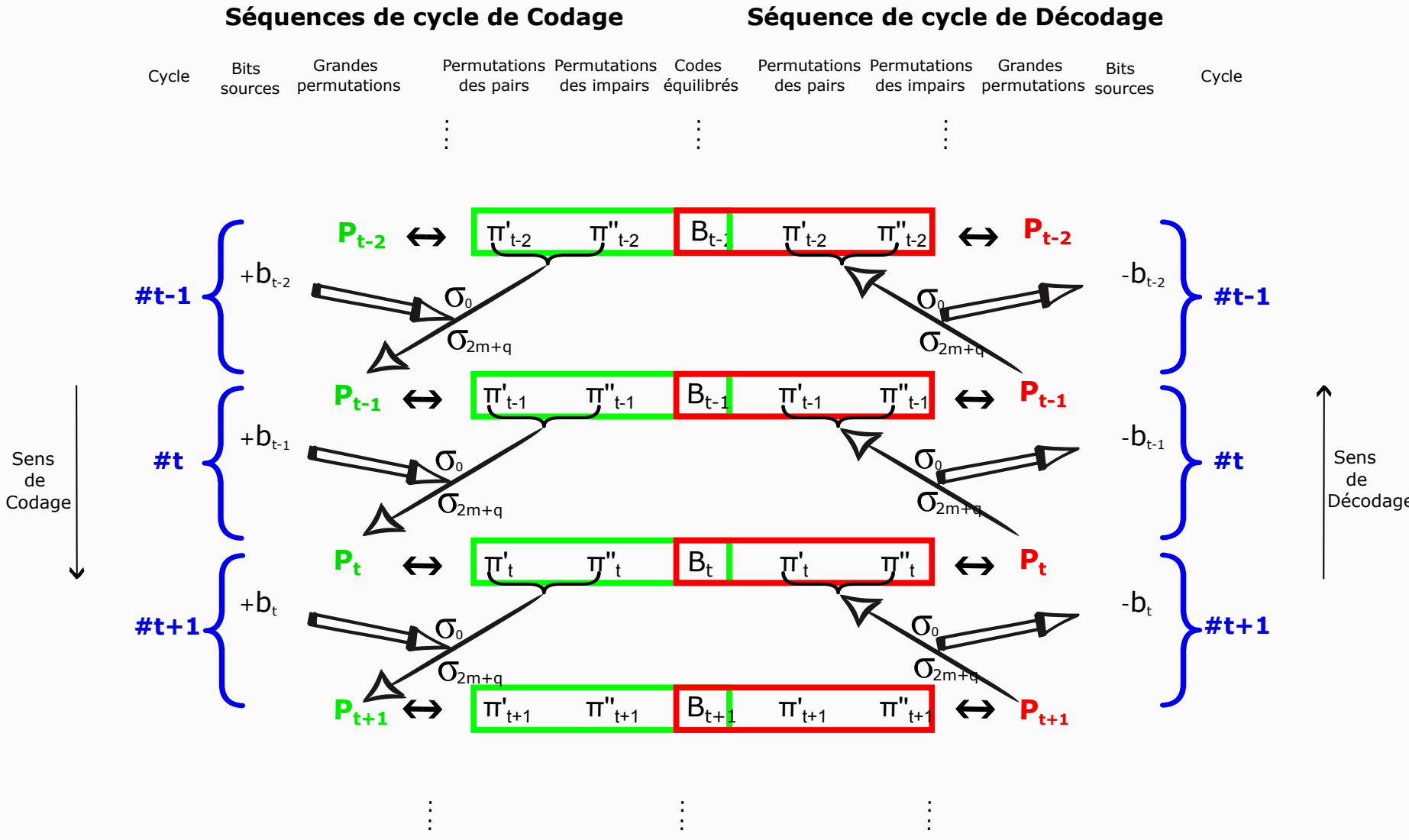
- Une programmation $\mathbb{P}$ consomme $\frac{m}{2}$ index $\iota_?$, $\frac{m}{2}$ index $\iota'_?$ et q bits $b_?$ frais et produit m index $H_?$ en $2 \times m + q$ d'opérations.
- Soit σ_0 la taille initiale de $\mathbb{P}$. Grace à cette information, nous pouvons savoir la taille de la mémoire σ_i après l'exécution des i premières instructions de $\mathbb{P}$. En particulier, la taille de mémoire à la fin de l'exécution de $\mathbb{P}$ est $\sigma_{2\times m+q}$.

Validité d'une programmation

Conditions de validité d'une Programmation $\mathbb{P}$

- Pour contrôler la mémoire dans chaque étape de la programmation, nous avons imposé une limite à la taille de la mémoire appelé Ω
- La programmation doit vérifier les deux conditions suivantes : $\forall\ 0 \leq i \leq 2 \times m + q.\ \sigma_i \leq \Omega$ et $\sigma_{2\times m+q} \leq \sigma_0$
- Par conséquence, $q \leq \log_2 \binom{m}{m/2}$, et Ω doit être suffisamment élevé pour pouvoir consommer la totalité des q bits.

Cycles de codage et de décodage



Cycle de codage et de décodage

Initialisation et terminaison

Initialisation et terminaison de $\mathbb{P}$

- La mémoire de Pacman est initialisée à la valeur 1 et la taille $\sigma_{2\times m+q}$. Les permutations π_0 et π'_0 sont initialisées à la permutation d'identité, $(1, ..., m/2)$.
- Le décodeur doit recevoir les données suivantes :
  - Un bloc de taille $\log_2(q)$ pour le nombre des bits comblés.
  - Un bloc de taille $\log_2(\Omega)$ qui contient la valeur finale de $\sigma_{2\times m+q}$.
  - Un bloc de taille $\log_2(m/2)$ pour chaqun des éléments des deux petites permutations finales.

La conception d'une programmation valide optimale

Définitions d'optimalité

- Nous considérons que le problème de la conception d'une programmation optimal est NP-difficile.
- Nous avons adopté deux définitions d'optimalité :
  - Def 1 : Choisir $q \leq \log_2 \binom{m}{m/2}$ et essayer de trouver Ω_{min}.
  - Def 2 : Choisir Ω et $\sigma_0 \leq \Omega$ et trouver Q_{max}.

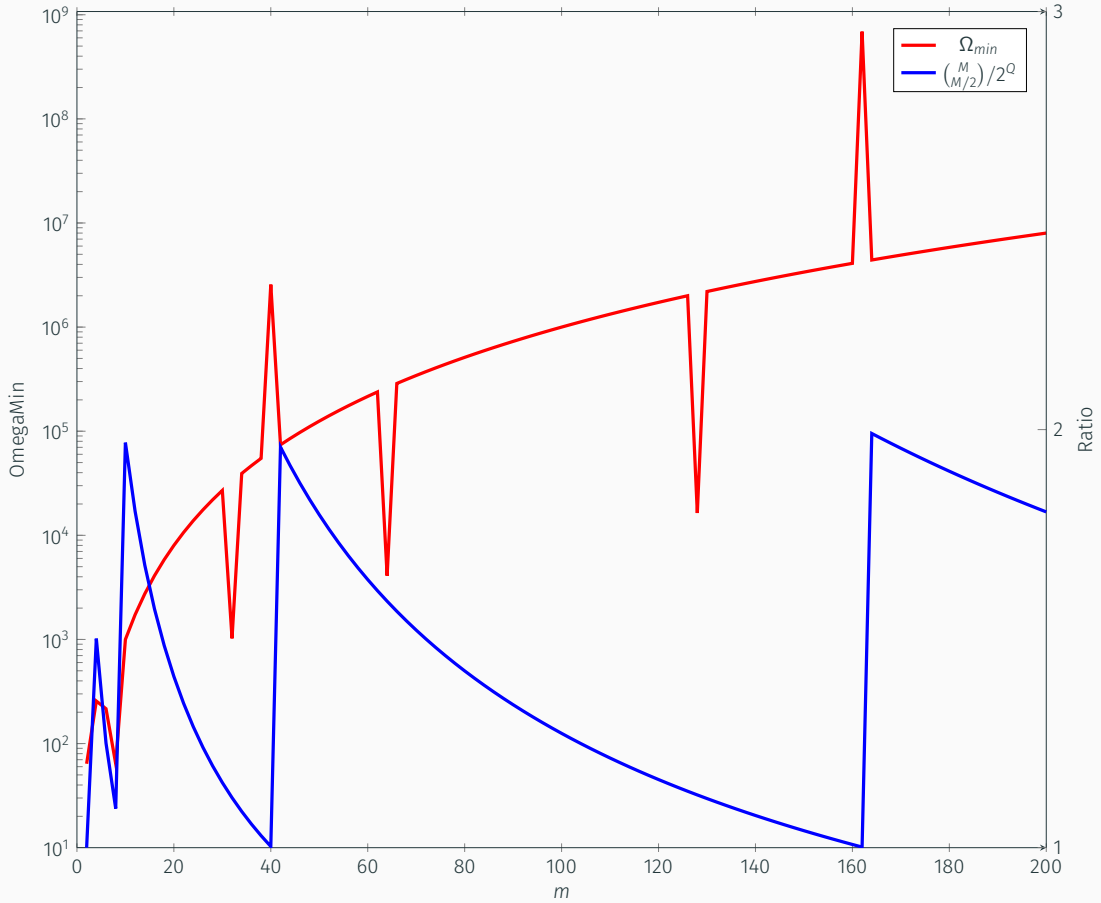
heuristiques adoptées

- Pour q, Ω et σ_0 choisies, nous créons en pire des cas une programmation valide.
- Pour une programmation en pire des cas valide créée à priori, nous essayons de faire l'optimisation en interchangeant les opérations pour diminuer la taille de la mémoire.

Résultats

Mode de redondance minimale

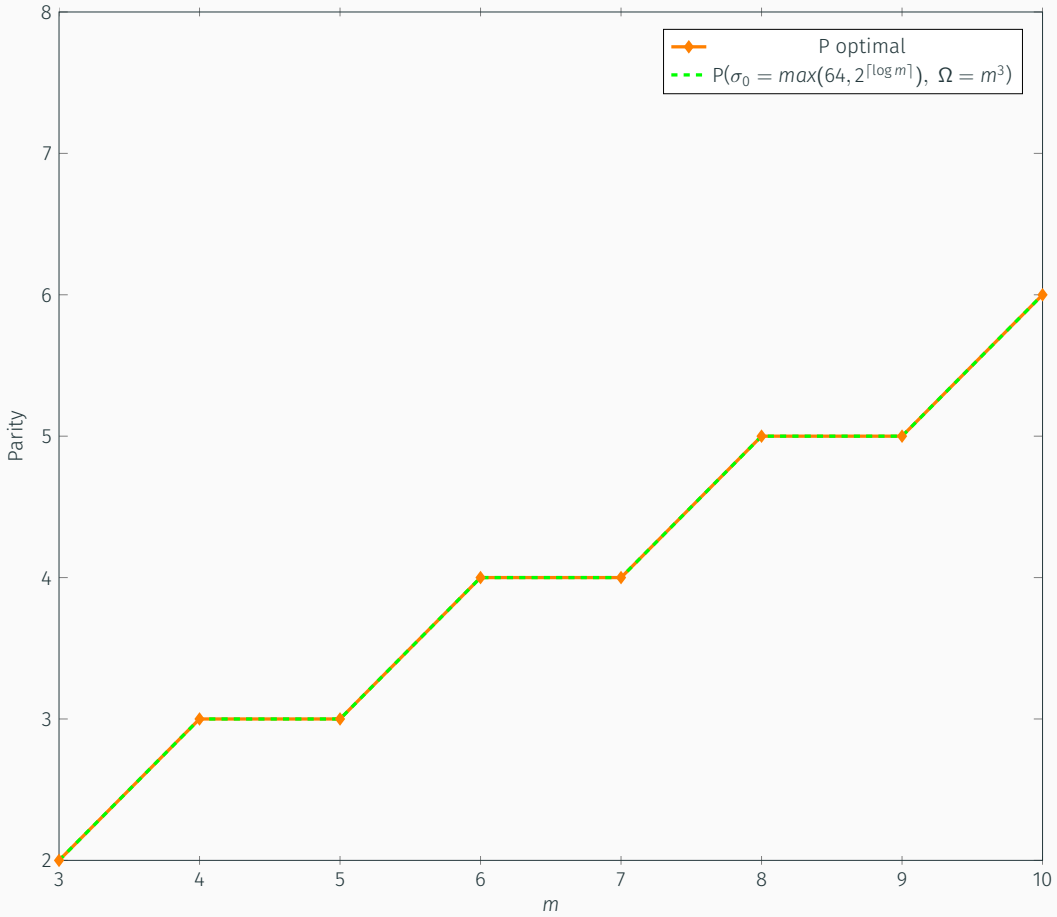
- Mode 1 : Pour $q = \left\lfloor \log_2 \binom{m}{m/2} \right\rfloor$ et $\sigma_0 = max(64, 2^{\lceil \log_2 M \rceil})$, nous varions M et nous traçons Ω_{min}.



la variation de Ω_{min} et du rapport par rapport $\binom{M}{M/2}/2^Q$ pour différentes valeur de m

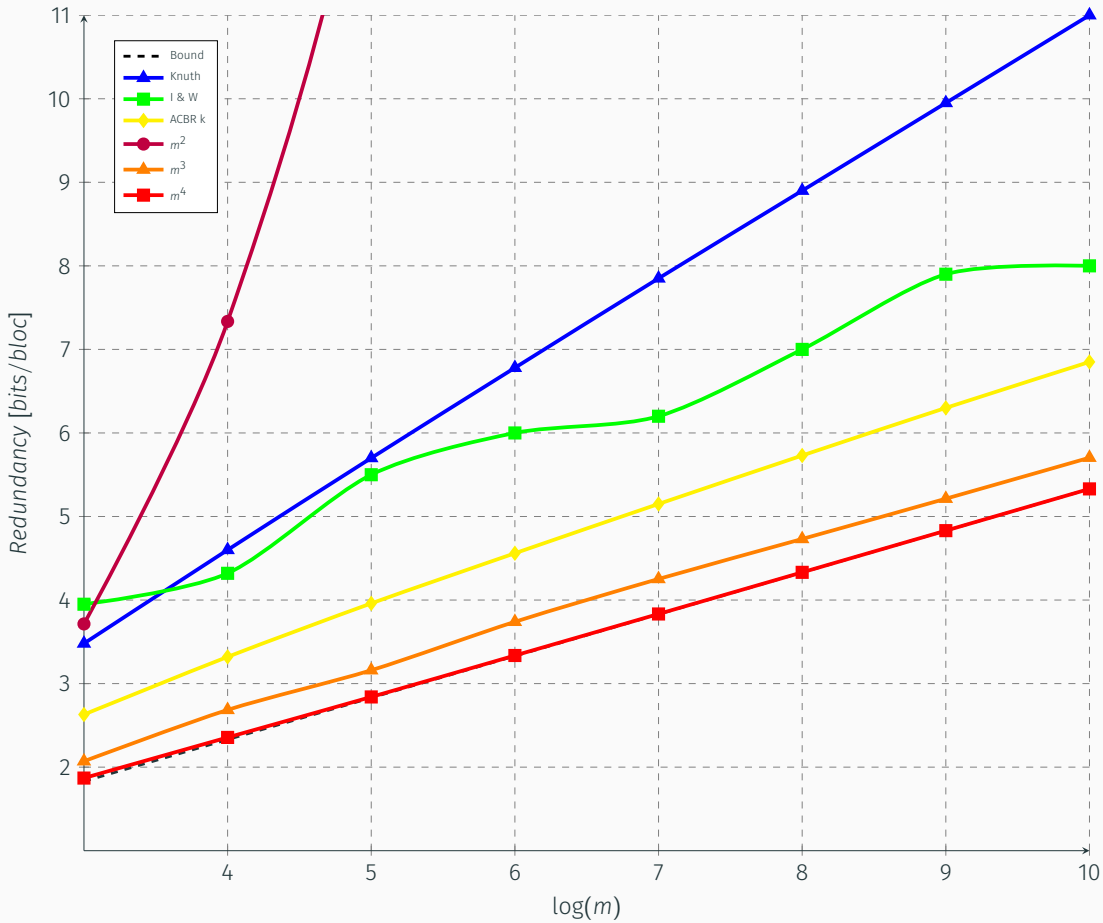
Mode pour les nombres entiers à précision limitée

- Mode 2 : Pour $\sigma_0 = max(64, 2^{\lceil \log_2 M \rceil})$ et $\Omega = M^k$ où $k \geq 2$, nous varions M et nous traçons la parité pour chaque valeur de Ω.



Le tracé de la parité

Calcul théorique des bornes supérieures de la redondance



Le tracé des bornes de la redondance pour différentes valeur de m et de Ω

Conclusion

Résumé

Les points importants

- **Problématique :** Comment utiliser efficacement les codes binaires équilibrés ?
- **Réalisation :** Nous avons conçu une technique basée sur les permutations, le caractère du jeux vidéo PAC-MAN et les entiers à précision limitée pour créer les code équilibrés.
- **Résultat :**
  - La redondance est nettement meilleure que celles des travaux antérieurs.
  - Les calculs dans notre technique ne sont pas coûteux et la complexité temporelle et spatiale pour le codage ou le décodage d'un bloc est linéaire.
- **Point faible :** le codage/décodage se font pas à la volée.

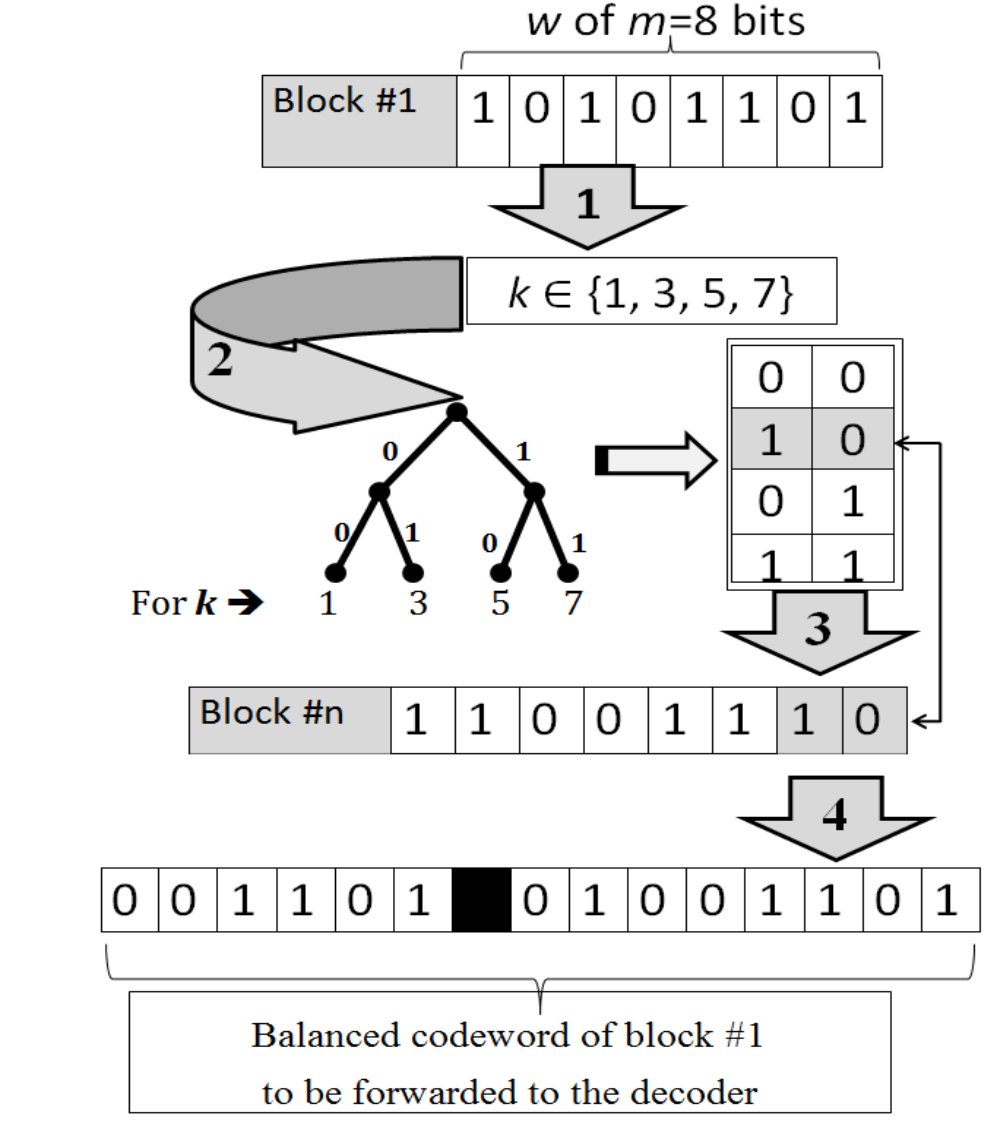
Travaux futurs

Perspectives

- Améliorer l'initialisation et la terminaison.
- Un mot de code équilibré de longueur m est capable d'incorporer $\log_2 \binom{m}{m/2}$ bits source et non pas seulement $\left\lfloor \log_2 \binom{m}{m/2} \right\rfloor$ bits. Nous pouvons aller au-delà des limites des nombres entiers et résoudre le problème avec des nombres rationnels.

Questions?

Backup slides



Les principales étapes du recyclage des bits pour l'algorithme de Knuth.

References I


T. Cover.
Enumerative source encoding.
IEEE Transactions on Information Theory, 19(1):73–77, 1973.


K. A. S. Immink and J. H. Weber.
Very efficient balanced codes.
Selected Areas in Communications, IEEE Journal on, 28(2):188–192, 2010.


D. E. Knuth.
Efficient balanced codes.
Information Theory, IEEE Transactions on, 32(1):51–53, 1986.


E. Sperner.
Ein satz über untermengen einer endlichen menge.
Mathematische Zeitschrift, 27(1):544–548, 1928.