Rapport Projet de Synthèse

Marius AMBAYRAC et Andrés Garcia

2018

Table des matières

1 Introduction

2 Partie Théorique

2.1 Définitions essentielles

2.2 Inégalité de Kraft-McMillan

2.3 Bornes sur la longueur moyenne attendue d'un code

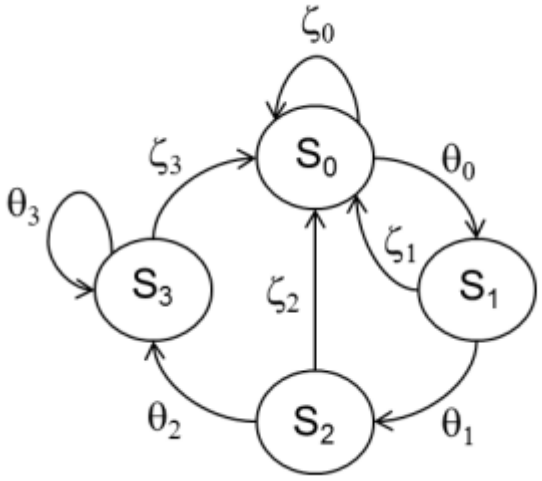
2.4 Optimalité du Codage de Huffmann

2.5 Limitations du Codage de Huffmann

3 Partie pratique

3.1 Source binaire sans mémoire

3.2 Source binaire Markovienne



4 Bibliographie

5 Annexe

5.1 Petites Fonctions

5.1.1 Calcul de l'entropie

```
            def entropie(tab_proba):
'''Cette fonction calcule l'entropie du tableau de
  probabilitÃ© fourni en entrÃ©e sous la forme de tuple
  (frequence, motif)'''
somme = 0
for (pi,elm) in tab_proba:
    somme += pi*log(pi,2)
return -somme
```

5.1.2 Longueur moyenne d'un codage

```
            def longueur_moyenne(source,codage):
'''Cette fonction calcule la longueur moyenne d'un codage.
   Elle prend en entrÃ©e une liste de tuple (frequence,
  motif) et une autre liste de tuple (code, motif).'''
long = 0
for (freq,motif) in source:
    for (code,motifBis) in codage :
        if motif==motifBis :
            long += len(code)*freq
return long
```

5.2 Codage de Hamming

```
    def Huffman(tab_proba):
"""Cette fonction prend en entrÃ©e une liste de tuple (
  frequence,motif)
et retourne une liste de tuple (code, motif)"""
m = len(tab_proba)
C = [("0","Initial")]*m
                                                    #C
```

```
    est le tableau de tuple que l'on renverra, il est
    composÃ© de (code, motifACoder)
  if m == 2 :

    #Cas de base de notre algorithme rÃ©cursif
      C[0],C[1] = ("0",tab_proba[0][1]),("1",tab_proba[1][1])
  else :
      tab_proba.sort(reverse = True)
                                                    #On trie
        la liste des proba dans le sens dÃ©croissant
      temp = Huffman(tab_proba[0:-2] + [(tab_proba[-2][0]+
        tab_proba[-1][0],"new")]) #On applique le procÃ©dÃ©
         sur une liste de taille m-1
      indice = recup(temp,"new")
      #On regarde ou l'Ã©lÃ©ment "spÃ©cial" s'est
        positionnÃ© suite au tri
      temp.append(temp[indice])
      temp.pop(indice)

        #On l'enlÃ¨ve pour le repositionner Ã  la fin de la
        liste
      for i in range(m-2):

        #On rÃ©cupÃ¨re les codes pour les m-2 premiers
        motifs
          C[i] = temp[i]
      C[m-2] = (temp[-1][0]+"0",tab_proba[-2][1])
                                          #On crÃ©e les codes
        pour les 2 derniers motifs
      C[m-1] = (temp[-1][0] + "1",tab_proba[-1][1])
  return C
```