Une nouvelle approche pour l'analyse d'algorithme en moyenne: le cas de l'entropie de Shannon

Olivier Bodini, Julien David, Izabell Iskandar

16 novembre 2021

Analyse d'algorithme

Complexité en moyenne

- Une distribution est considérée sur l'ensemble des entrées de l'algorithme.
- On considère souvent la distribution uniforme, ou un ensemble de distributions qui incluent l'uniforme.
- Complément d'information à l'analyse dans le pire des cas

Analyse d'algorithme

- Le comportement de plusieurs algorithmes **en pratique** fut expliqué par l'analyse en moyenne
  - algorithmes de tri,
  - algorithmes de recherche,
  - algorithmes du texte,

On ne se limite pas à la distribution uniforme

Différents modèles, ou distributions, furent par exemple considérés pour :

- le calcul du pcgd [Lhote, Vallée et al.],
- algorithmes de minimisation d'automates [Bassino., D., Nicaud],
- le calcul des traverses minimales d'hypergraphes [D., Lhote, Mary, Rioult], dont l'algorithme de Hébert, Bretto, Crémilleux.

Analyse d'algorithmes

Lorsqu'on analyse un algorithme en moyenne :

- on identifie une propriété sur l'entrée, qui garantit une borne supérieure/inférieure sur la complexité de l'algorithme.
- on prouve que, pour une (famille de) distribution(s), la probabilité que la propriété soit vraie tend vers 0/1/une constante.

Certains résultats de complexité en moyenne sont en réalité des résultats de complexité générique.

Analyse d'algorithmes : frustrations

`Dans la réalité la distribution n'est pas uniforme, donc votre résultat ne sert à rien`

... est une remarque à laquelle il faut se préparer quand on présente un résultat en moyenne

Cas moyen VS ”Réalité”

- Si, lors de l’analyse, on identifie une propriété qui garantie le comportement de l’algorithme,
- si cette propriété est pertinente/présente dans le contexte dans lequel on applique l’algorithme
- alors l’analyse en moyenne est pertinente pour comprendre comment l’algorithme se comporte en pratique.
- Dans le cas contraire, le résultat est (malheureusement) inutile **pour ce contexte**.

Analyse ”réaliste” d’algorithmes

Différentes approches ont été faite pour obtenir des résultats plus réalistes.

- Towards a realistic analysis of the QuickSelect algorithm (Julien Clément, James Allen Fill, Thu Hien Nguyen Thi, Brigitte Vallée)
- Good Predictions Are Worth a Few Comparisons (Nicolas Auger, Cyril Nicaud, Carine Pivoteau)

Ce que je cherche

Ce qui compte, ce sont les **propriétés qui modifient le comportement de l'algorithme** :

- dans l'idéal, on aurait une propriété dont la valeur serait le paramètre d'une fonction continue de l'efficacité de l'algorithme, du pire au meilleur des cas.
- identifier **une** propriété ne signifie pas que l'on a parfaitement compris l'algorithme.

Objectifs

Obtenir des méta-théorèmes sur la complexité moyenne des algorithmes :

- pour toute distribution de probabilité qui garantisse la valeur (moyenne) d'une propriété, on obtient la complexité moyenne de l'algorithme.

Pourrait être utile en milieu industriel : il peut être simple de mesurer la valeur d'une propriété sur leur données.

Analyse d'algorithmes : frustration

Ces résultats sont difficiles à obtenir :

- Identifier une propriété importante peut prendre un temps considérable (et une fois explicité, elle paraît évidente).

Analyse d'algorithmes : comment réduire la difficulté ?

Un outil aidant à l'analyse d'algorithme est la génération aléatoire :

- cela permet d'obtenir facilement des conjectures solides sur la complexité moyenne,
- mais cela nous aide rarement à identifier la propriété qui modifie le comportement de l'algorithme.
  - si le générateur produit des objets dont la propriété déterminante pour le comportement de l'algorithme ne varie pas ?
  - C'est souvent le cas avec des générateurs aléatoires uniformes : les objets engendrés ont une forme limite.

Sérendipité

La sérendipité est le fait de trouver des informations inattendues mais pertinentes.

Génération aléatoire et sérendipité

On souhaite :

- engendrer des objets dont les propriétés sont variées
- avoir à développer un minimum de générateurs
- ne pas contraindre les propriétés de l'objet **a priori**
- limiter le nombre de paramètres, qui viendraient compliquer l'analyse par la suite.

Analyse d'algorithmes : solution partielle

- Utiliser plusieurs générateurs qui produisent des objets avec différentes propriétés
  Exemple : le modèle d'Erdös-Renyi pour les graphes aléatoires.

Génération aléatoire : proposition

Idée : fabriquer un **générateur aléatoire de générateurs aléatoires**.
Plus précisément

- On veut engendrer aléatoirement des distributions de probabilité, dont l'entropie de Shannon est égale à une valeur fixée t.
- Créer un générateur aléatoire qui engendre des objets de taille n d'après cette distribution.

Exemples

Cette idée est valable si le générateur aléatoire produit des séquences :

- de variables aléatoires i.i.d.
- tirées dans un ensemble fini de k valeurs.

Par exemple :

- Séquences aléatoires dans un alphabet fini,
- Processus de Galton Watson,
- Matrices/Graphes/Hypergraphes aléatoires,
- Automates déterministes complets.

Entropie de Shannon

Soit une distribution de probabilité π sur k évènements, où le i-ème évènement a une probabilité π_i d'être tirée.

L'entropie de Shannon, notée $H(\pi)$, est définie par :

$$H(\pi) = \sum_{i=1}^{k} \pi_i \cdot log_2(\frac{1}{\pi_i})$$

On a :

$$0 \leq H(\pi) \leq log_2(k)$$

Génération aléatoire : distribution à entropie fixée

Pour une entropie t fixée, on veut donc engendrer aléatoirement un vecteur $(p_1, \ldots, p_k)$ tel que :

$$\begin{cases} \sum_{i=0}^{k} p_i = 1 \\ H(p_1, \ldots, p_k) = t \end{cases}$$

Et utiliser ensuite cette distribution afin de faire de la génération aléatoire.

Pourquoi l'entropie de Shannon ?

- C'est une généralisation de la distribution uniforme (entropie maximale).
- Quand l'entropie décroit, des motifs tendent à se répéter,
- Dans d'autres cas, l'entropie est elle-même la propriété déterminante du comportement de l'algorithme.

Méthode probabiliste

Méthode probabiliste

Algorithme 1 : Méthode probabiliste

Entrées : k évènements, une valeur t
Sorties : une distribution aléatoire d'entropie t
Faire
  $(x_1, \ldots, x_k) \leftarrow$ variables aléatoires selon une loi exponentielle de paramètre 1;
  $s \leftarrow \sum_{i=1}^{k} x_i$;
  $\pi \leftarrow (x_1/s, \ldots, x_k/s)$;
Tant Que $H(\pi) \neq t$;
retourner π;

Méthode probabiliste - Densité

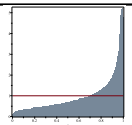
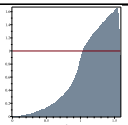
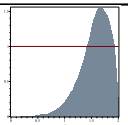
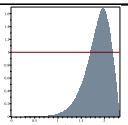
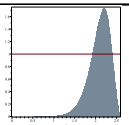
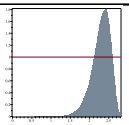
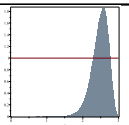
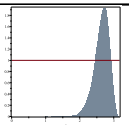
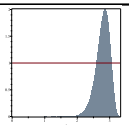
k	2	3	4	5	6	7	8	9	10
$\mathbb{E}(t) \approx$	0.721	1.202	1.563	1.852	2.092	2.299	2.478	2.639	2.782
$\sigma(t) \approx$	0.271	0.284	0.278	0.265	0.253	0.242	0.232	0.221	0.215
Density									

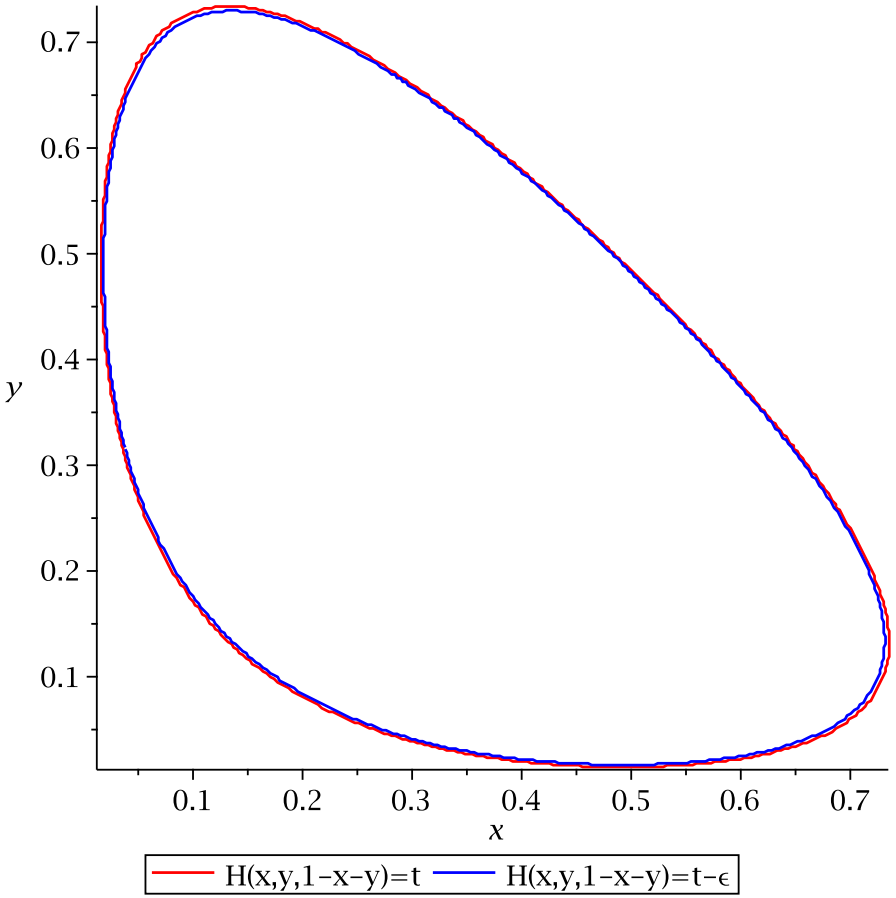
FIGURE – Entropie moyenne, écart-type et courbe de la densité (k de 2 à 10). La ligne rouge est la densité 1 (uniforme).

Méthode probabiliste - Rejet

- Si on considère un rejet du type :

$$t - \epsilon < H(x_1/s, \ldots, x_k/s) < t + \epsilon$$

- la largeur de l'intervalle obtenu n'est pas constante.

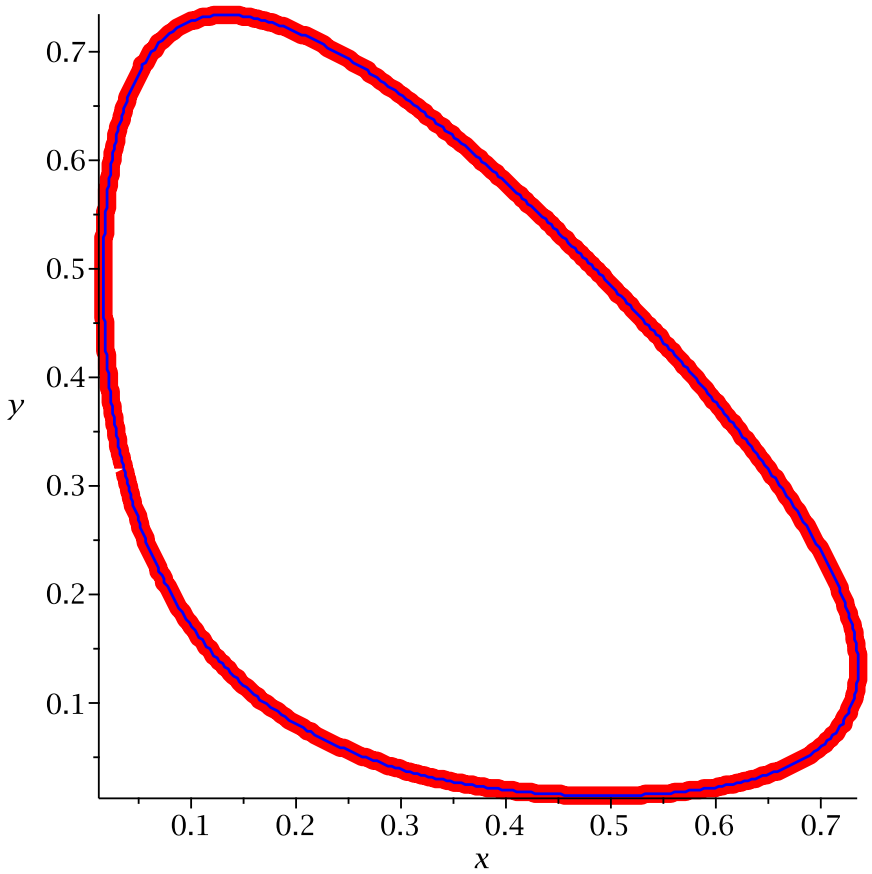


Méthode probabiliste - Rejet

Soit la surface

$$\{(x_1, \ldots, x_k); H(x_1/s, \ldots, x_k/s) = t\}$$

- Pour chaque point p de cette surface, on conserve les points $p + x\mathbf{n}$ où $\mathbf{n}$ est le vecteur normal à la surface et $x \in [-\epsilon, \epsilon]$.



Méthode probabiliste - Amélioration

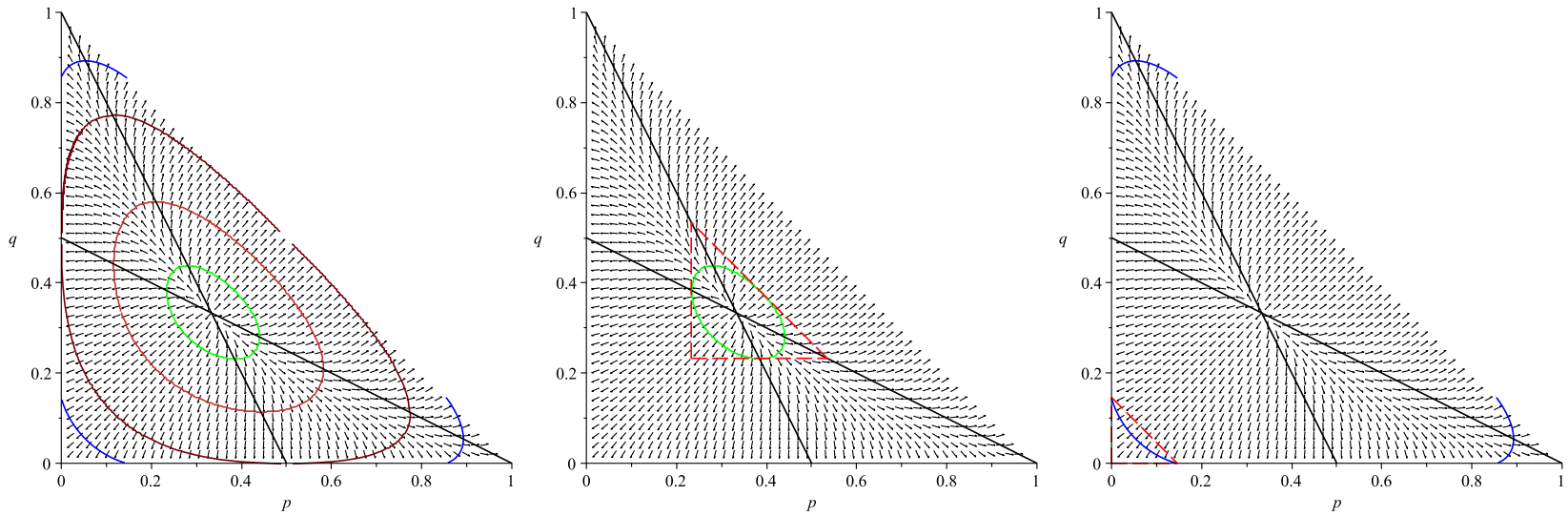


FIGURE – On the left, for $k = 3$, the plots of the points $(p, q, 1 - p - q)$ of entropy s for $s = 0.6$ in blue, $s = 1$ in red, $s = 1.4$ in orange and $s = 1.55$ in green. In the middle, in dashed red, the restricted area where the points are uniformly drawn for rejection optimization in the case of large entropy. On the right, the same thing but for small entropy.

Chaîne de Markov

L'idée

Pour que cette méthode fonctionne nous avons besoin :

- Que l'ensemble des solutions soit fini.
- De définir des générateurs ad-hoc pour $k = 2$ et $k = 3$
  - l'algorithme pour $k = 3$ nous servira de pas dans la chaîne de markov.
  - l'algorithme pour $k = 3$ utilise l'algorithme pour $k = 2$
  - l'idée de l'algorithme pour $k = 2$ peut être généralisée pour obtenir une initialisation de la chaîne de Markov.
  - on généralise ces algorithmes afin de la somme des variables soit un $s \leq 1$ fixé.

Prérequis

Pour cette méthode, on considère qu'une probabilité a une valeur minimale ε et que toute valeur est un multiple de ε.
Deux valeurs t et t' sont ε**-équivalentes**, noté $t \sim_\varepsilon t'$, s'il existe deux distributions π, π' qui ne diffèrent que sur deux variables d'indices i et j telles que :

- $\pi_i = \pi'_i + \varepsilon$, $\pi_j = \pi'_j - \varepsilon$
- $H(\pi) = t < t' \leq H(\pi')$.

Distribution de probabilité discrète à entropie fixée

Soit $\mathcal{D}(s, k, t, \varepsilon)$ un ensemble de k-tuples π tels que :

- $\sum_{i=1}^{k} \pi_i = s$
- toute probabilité est un multiple de ε.
- $H(\pi) \sim_\varepsilon t$

On veut obtenir un générateur aléatoire uniforme pour $\mathcal{D}(s, k, t, \varepsilon)$.

Remarque

Soit un tuple $(x_1, \ldots, x_k)$ tel que $\sum_{i=1}^{k} i = s \leq 1$.

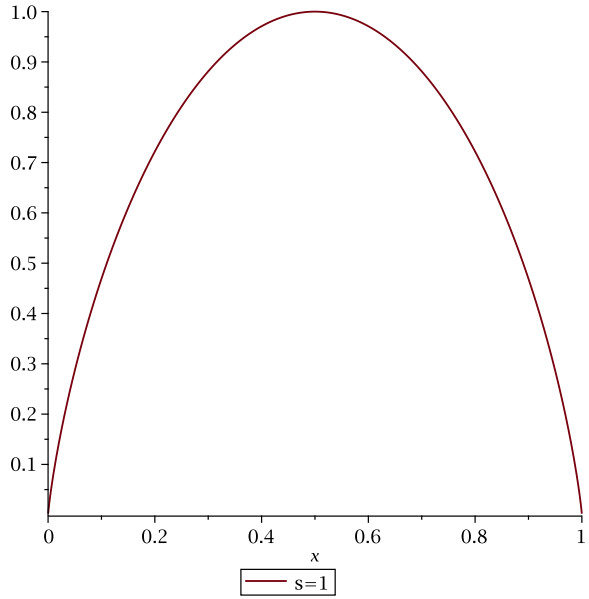
- $H(x_1, \ldots, x_k)$ est maximale en $H(\frac{s}{k}, \ldots, \frac{s}{k})$
- $H(x_1, \ldots, x_k)$ est minimale en $H(s - \varepsilon, \varepsilon, \ldots, \varepsilon)$

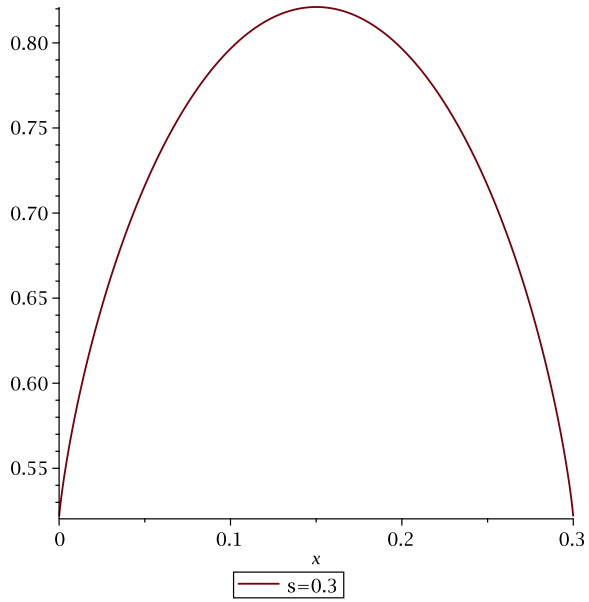
Algorithme 2 : *RandomDistributionK2*(s, t)

Entrées : Two values s and t such that
$H(s - \varepsilon, \varepsilon) \leq t \leq H(\frac{s}{2}, \frac{s}{2})$

Sorties : A couple of values $(x, s - x)$ s.t.
$H(x, s - x) \sim t$

$x \leftarrow \frac{s}{2}$;
$\delta \leftarrow \frac{s}{2}$;
$t' \leftarrow H(x, s - x)$;
tant que $t' \neq t$ *or* $\delta \geq \varepsilon$ **faire**
  si $t' < t$ **alors**
    $x \leftarrow x + \delta$;
  fin
  sinon
    $x \leftarrow x - \delta$;
  fin
  $\delta \leftarrow \frac{\delta}{2}$;
  $t' \leftarrow H(x, s - x)$;
fin
return $(x, s - x)$ *with probability* $\frac{1}{2}$ *and* $(s - x, x)$ *otherwise*





$k = 3$

L'idée :

- on veut tirer aléatoirement dans $\mathcal{D}(s, 3, t, \varepsilon)$
- on tire une valeur x dans un intervalle valable.
- on tire (x, y) dans $\mathcal{D}(s - x, 2, t - H(x), \varepsilon)$

$k = 3$

Algorithme 3 : RandomVectorK3

Input : A probability distribution π over k events such that $H(\pi) \sim t$
Output : A modified probability distribution π over k events
$(x, y, z) \leftarrow$ pick 3 events in a random order amongst the k events;
$s \leftarrow x + y + z$;
$x' \leftarrow$ random value in *Interval*;
$s' = s - x'$;
$t' \leftarrow t - H(x')$;
if $H(s' - \varepsilon, \varepsilon) \leq t' \leq H(\frac{s'}{2}, \frac{s'}{2})$ **then**
  $(y', z') \leftarrow$ *RandomDistributionK2*(s', t');
  if $y' \neq z'$ *or with probability* $\frac{1}{2}$ **then**
    $\pi \leftarrow$ replace (x, y, z) by (x', y', z') in π;
  end
  return π
end

$k = 3$

Rappel :

- $H(x_1, \ldots, x_k)$ est maximale en $H(\frac{s}{k}, \ldots, \frac{s}{k})$
- $H(x_1, \ldots, x_k)$ est minimale en $H(s - \varepsilon, \varepsilon, \ldots, \varepsilon)$

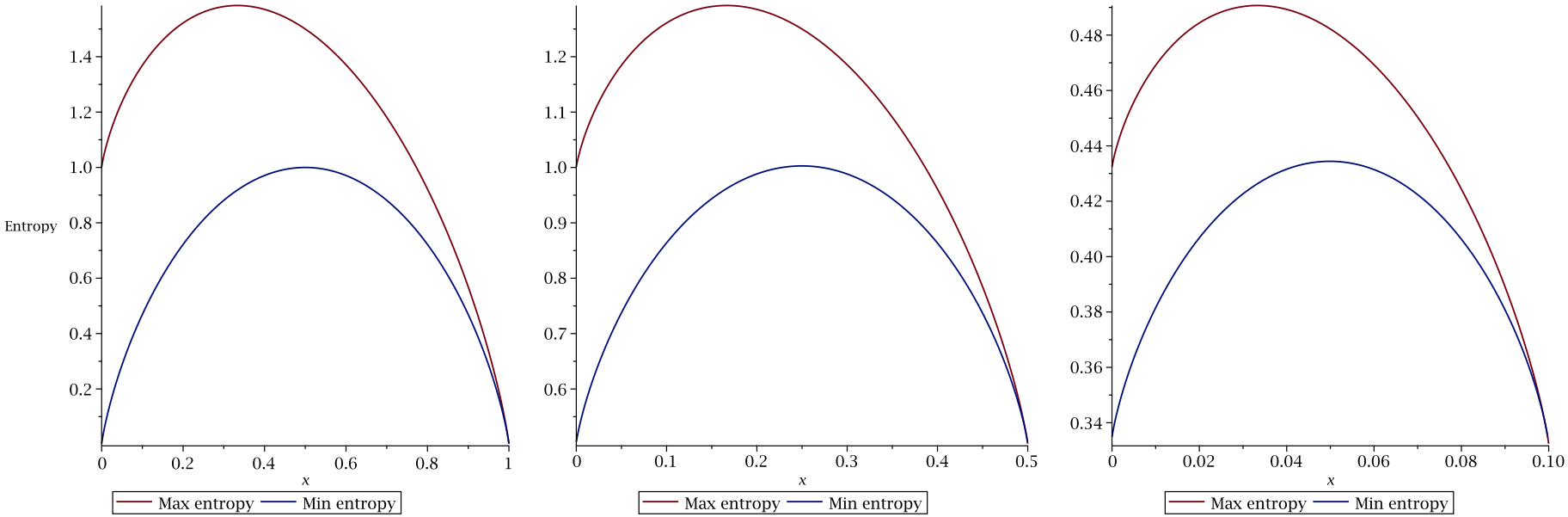


FIGURE – De gauche à droite : $s = 1$, $s = 0.5$, $s = 0.1$

$k = 3$, l'intervalle

Lemma

Given two values $t \in]0, \ldots, log_2(3)]$ *and* $s \in]0, \ldots, 1]$*, the system*

$$\begin{cases} H(x, y, z) = t \\ x + y + z = s \end{cases}$$

has a solution if and only if

$$x \in \begin{cases}]0, \ldots, minInv(s, t)] \cup [s - minInv(s, t), \ldots, maxInv_1(s, t)] \text{ if } t < H(\frac{s}{2}, \frac{s}{2}) \\ [maxInv_0(s, t), \ldots, maxInv_1(s, t)], \text{ otherwise.} \end{cases}$$

Générateur aléatoire

Algorithme 4 : Générateur à base de chaîne de Markov

Entrées : k évènements, une valeur t
Sorties : une distribution aléatoire de $\mathcal{D}(k, m, t)$
$\pi \leftarrow MarkovChainInitialState(k, t)$;
for $i \in \{1, \ldots, steps\}$ **do**
  $\pi \leftarrow RandomVectorK3(\pi, t)$;
end
return π

Retour à l'analyse d'algorithmes

Algorithme du texte :
`Entrée` : un texte de longueur n, un motif de longueur m, un alphabet k
`Sortie` : nombre d'occurrences du motif dans le texte.

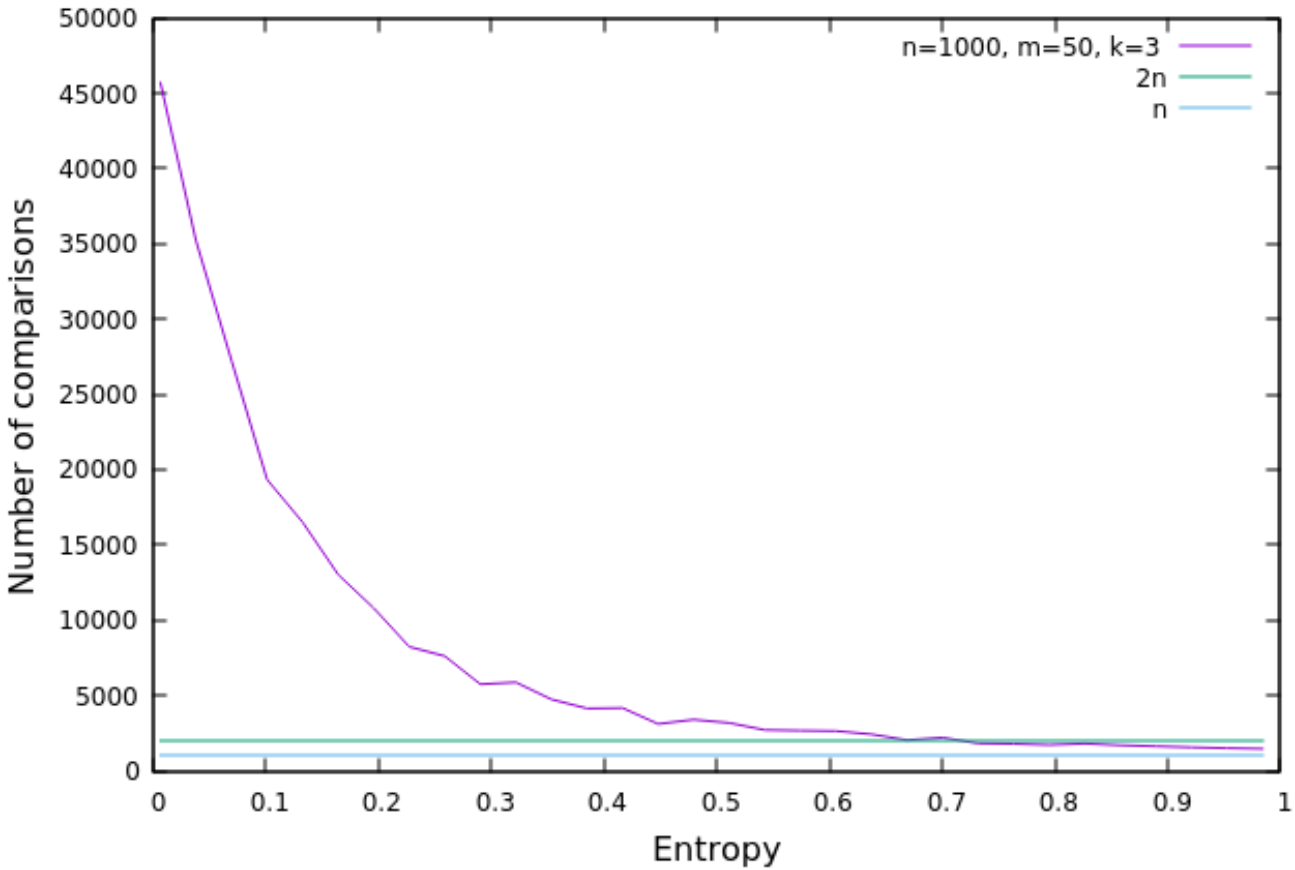
L'opération qui détermine la complexité est le nombre de comparaisons.

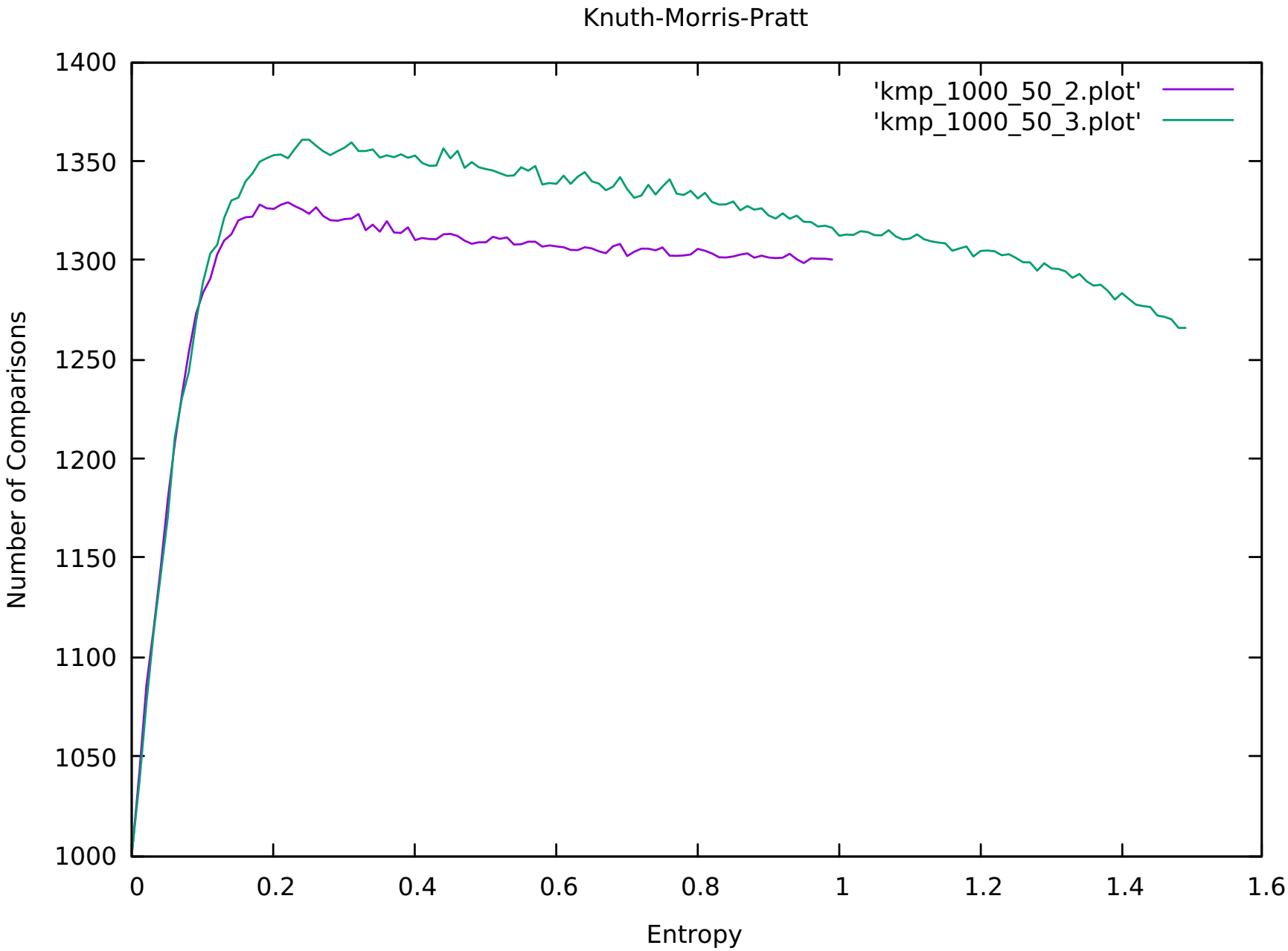
Retour à l'analyse d'algorithmes

Expériences sur 3 algorithmes (parmi une centaine) :

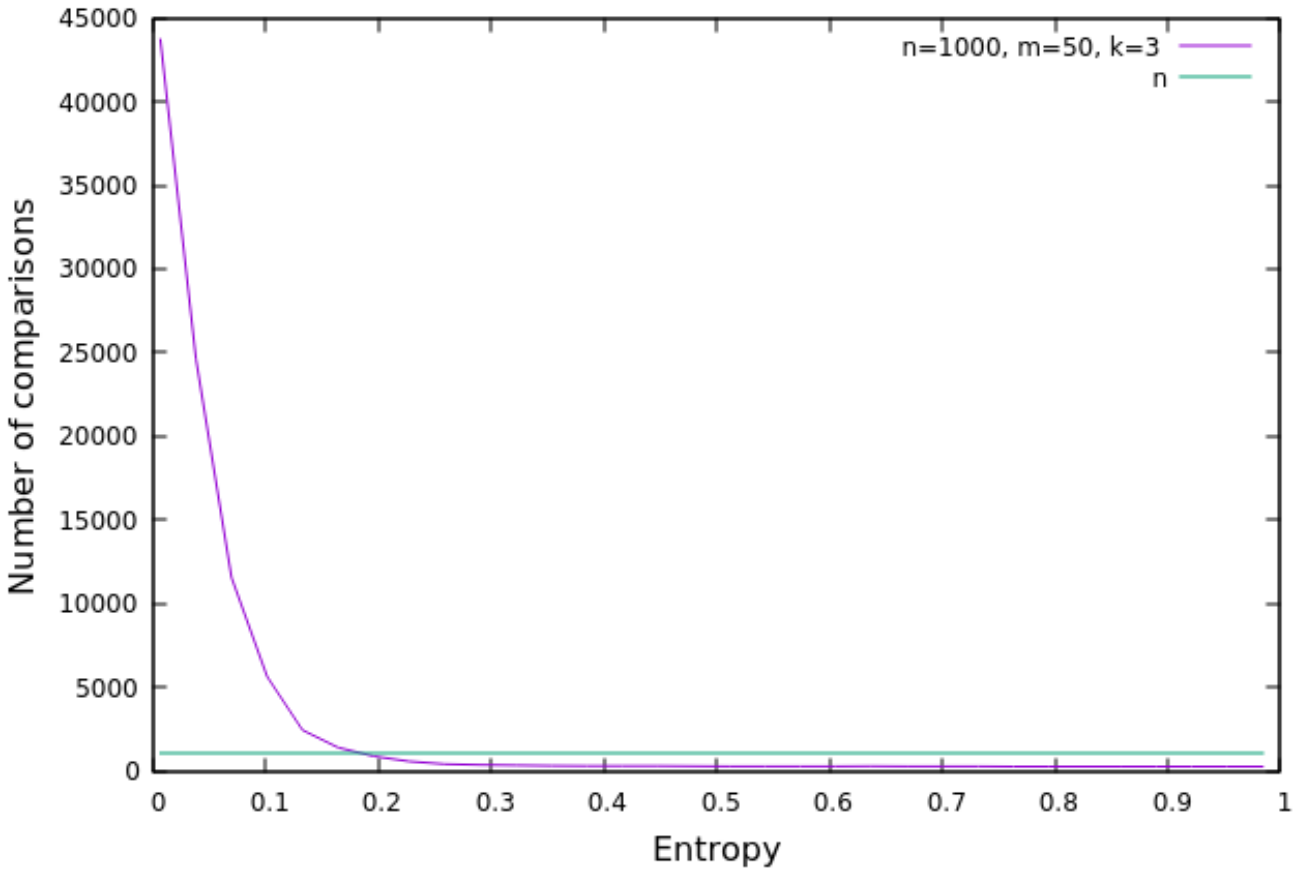
- **Naïf**
  - Pire Cas : $\Theta(nm)$
  - Meilleur Cas : $\Theta(n)$
  - Cas Moyen/Générique : $\Theta(n)$
- **Knuth-Morris-Pratt**
  - Pire Cas : $\Theta(n)$
  - Meilleur Cas : $\Theta(n)$
- **Boyer-Moore**
  - Pire Cas : $\Theta(nm)$
  - Meilleur Cas : $\Theta(\frac{n}{m})$
  - Cas Moyen : $\Theta(n)$ (the constant is less than 1).

Naive Algorithm




Knuth-Morris-Pratt
'kmp_1000_50_2.plot'
'kmp_1000_50_3.plot'
Number of Comparisons
Entropy
1400
1350
1300
1250
1200
1150
1100
1050
1000
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

Boyer-Moore Algorithm



Complexité moyenne

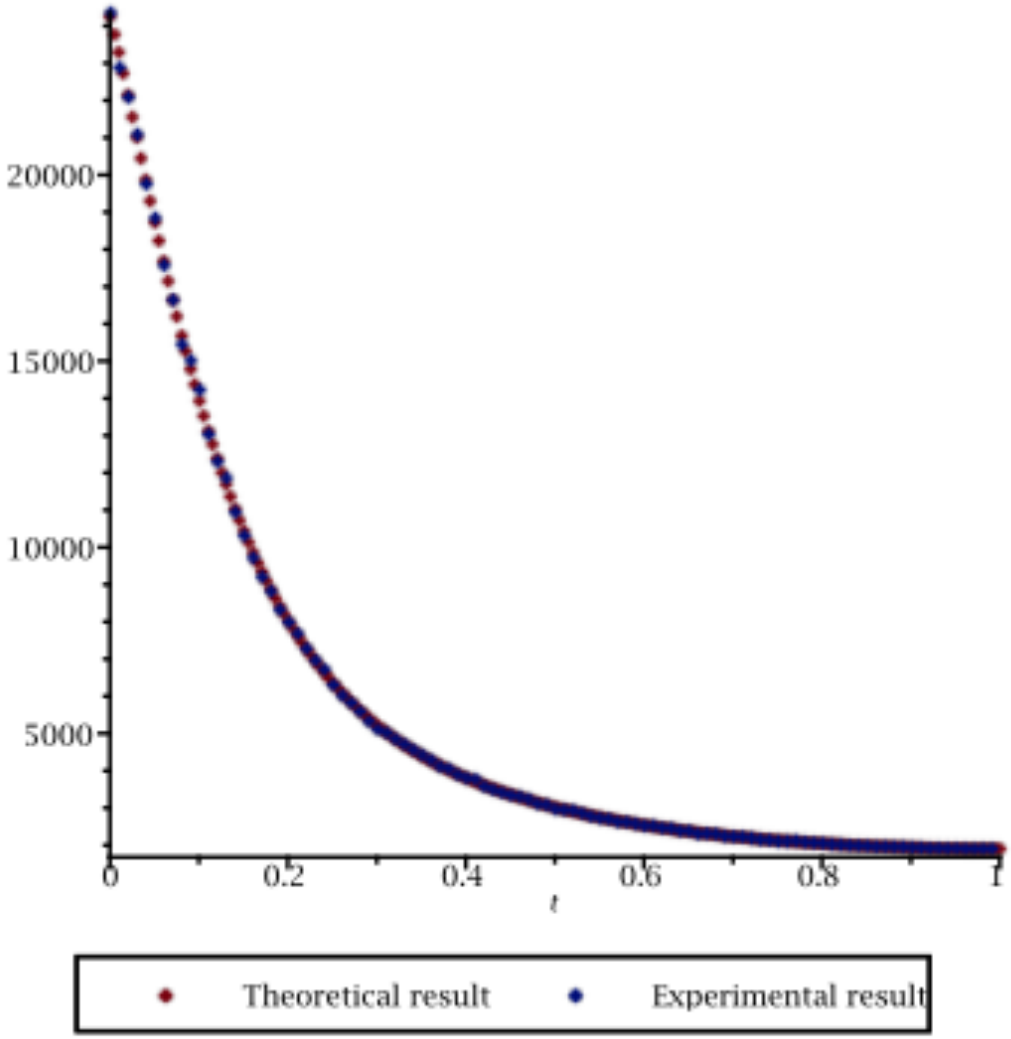
Theorem

Pour $k = 2$, la complexité moyenne de l'algorithme naïf sur un texte de longueur n et un motif de longueur m engendrés par des sources d'entropies t est

$$(n - m + 1)\frac{(\sum_{j=1}^{m} cp(t)^j + 1 - acp(t)^m)(\sum_{j=1}^{m} cpd(t)^j + 1 - cpd(t)^m)}{2}$$

où cp est la "collision probability" quand le texte et le motif partagent la même distribution et cpd quand elles diffèrent.

Complexité moyenne vs résultats expérimentaux



Algorithmes de recherche de séquences

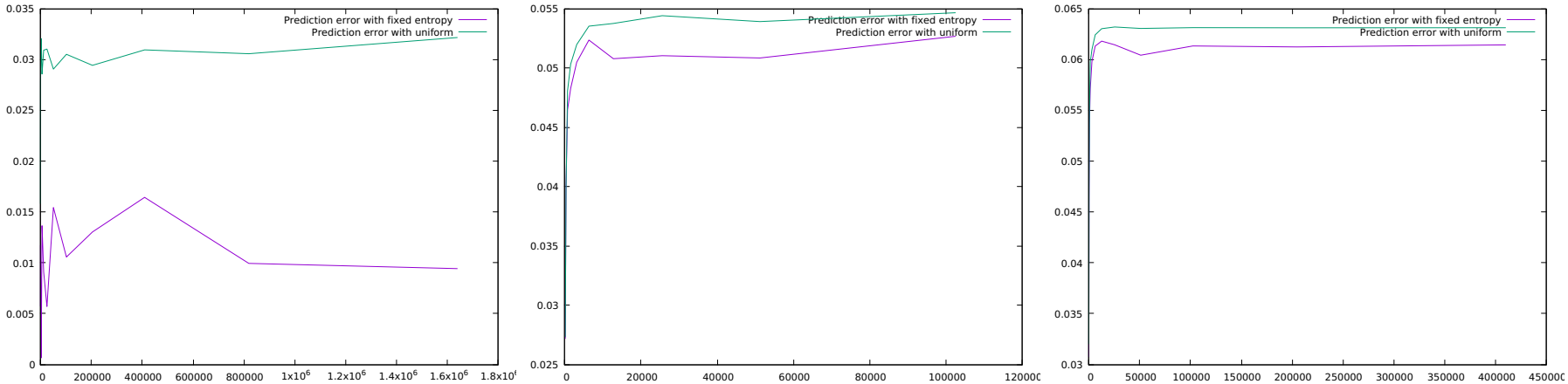
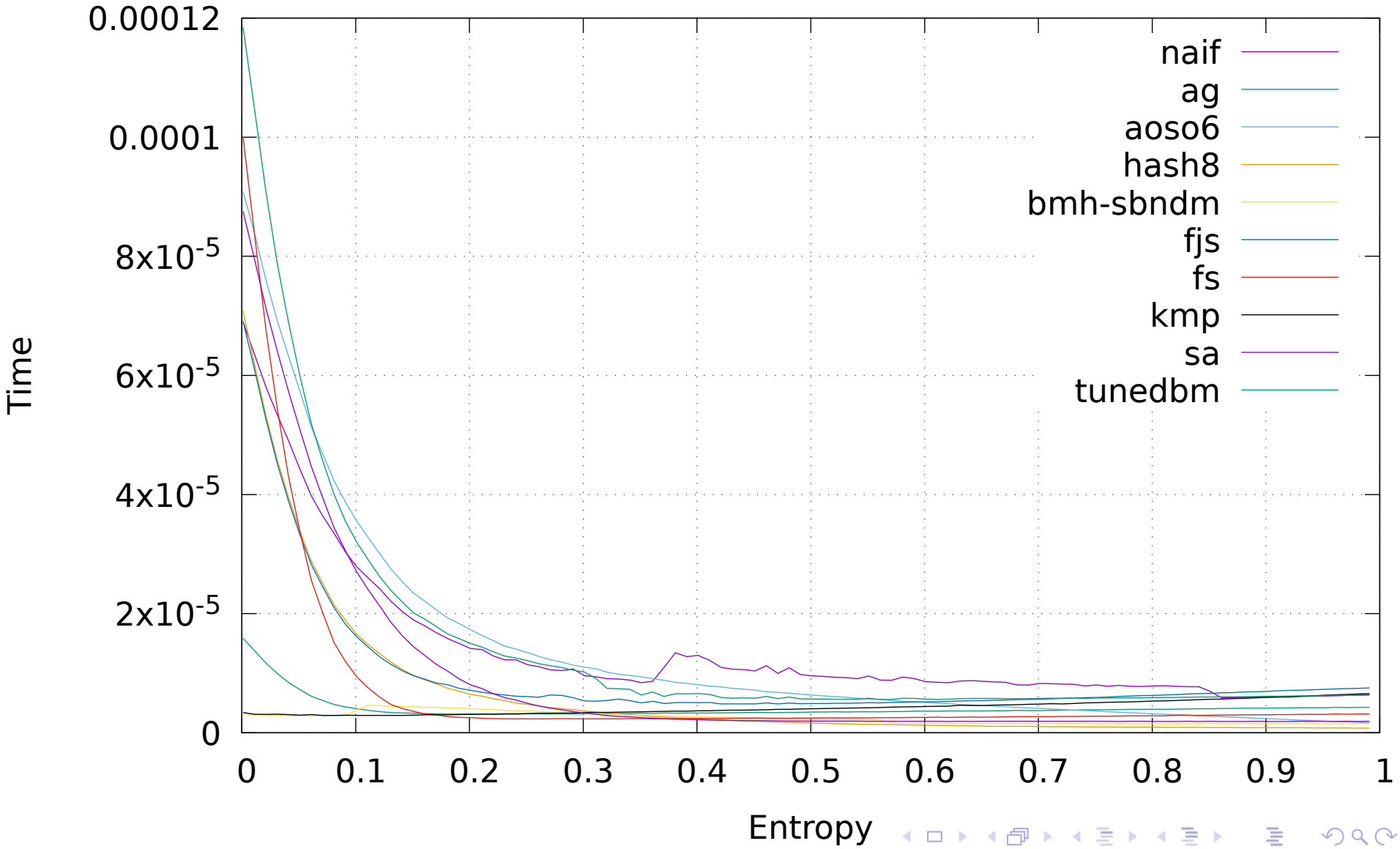


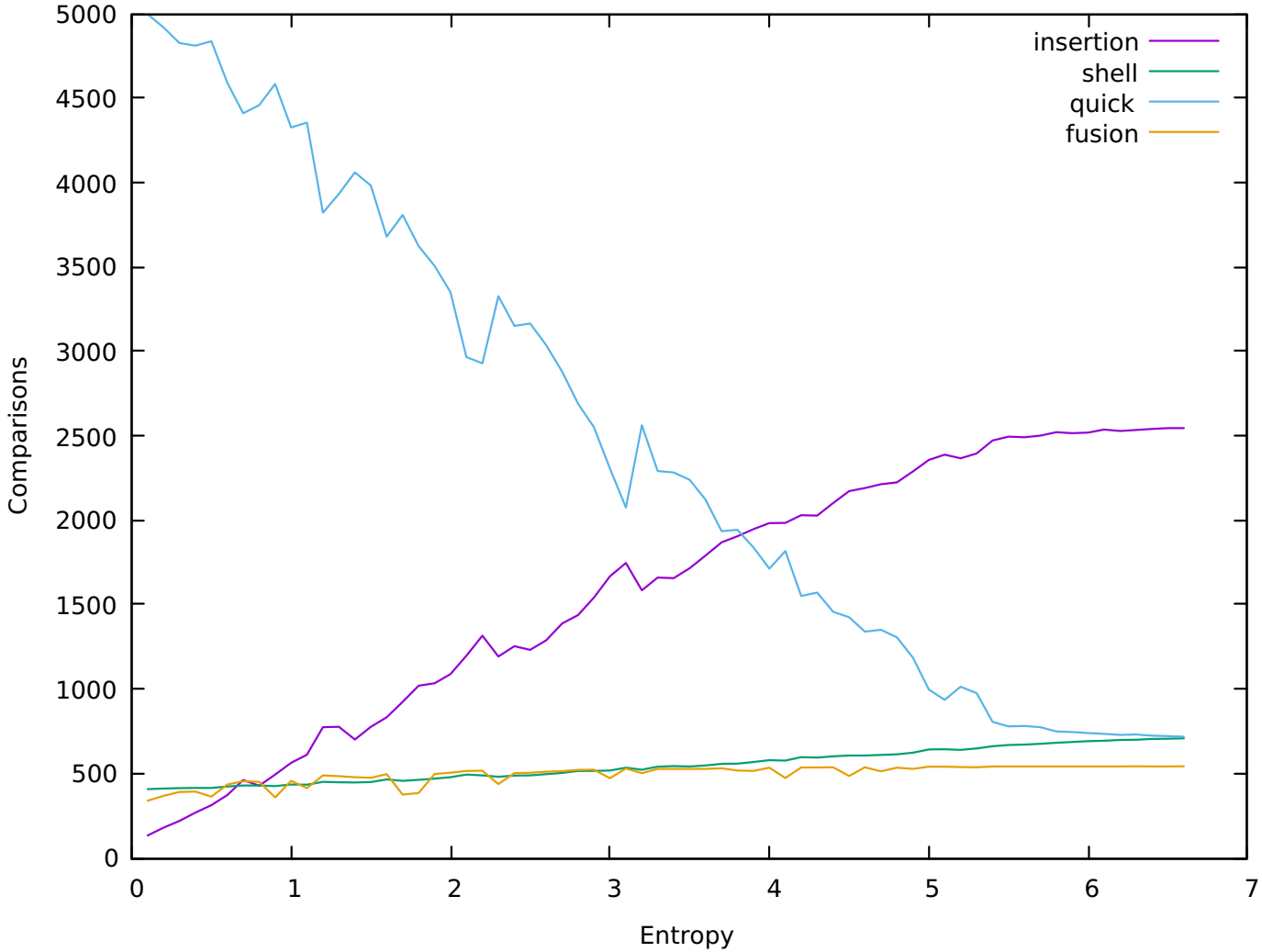
FIGURE – Ratio d'erreur de prédictions sur des séquences issues (de gauche à droite) d'un chromosome 21 ($k = 4$, $t = 1.35691$), de Hamlet ($k = 88$, $t = 3.0439$), du Tour du monde en 80 jours ($k = 78$, $t = 3.00892$).

Merci !

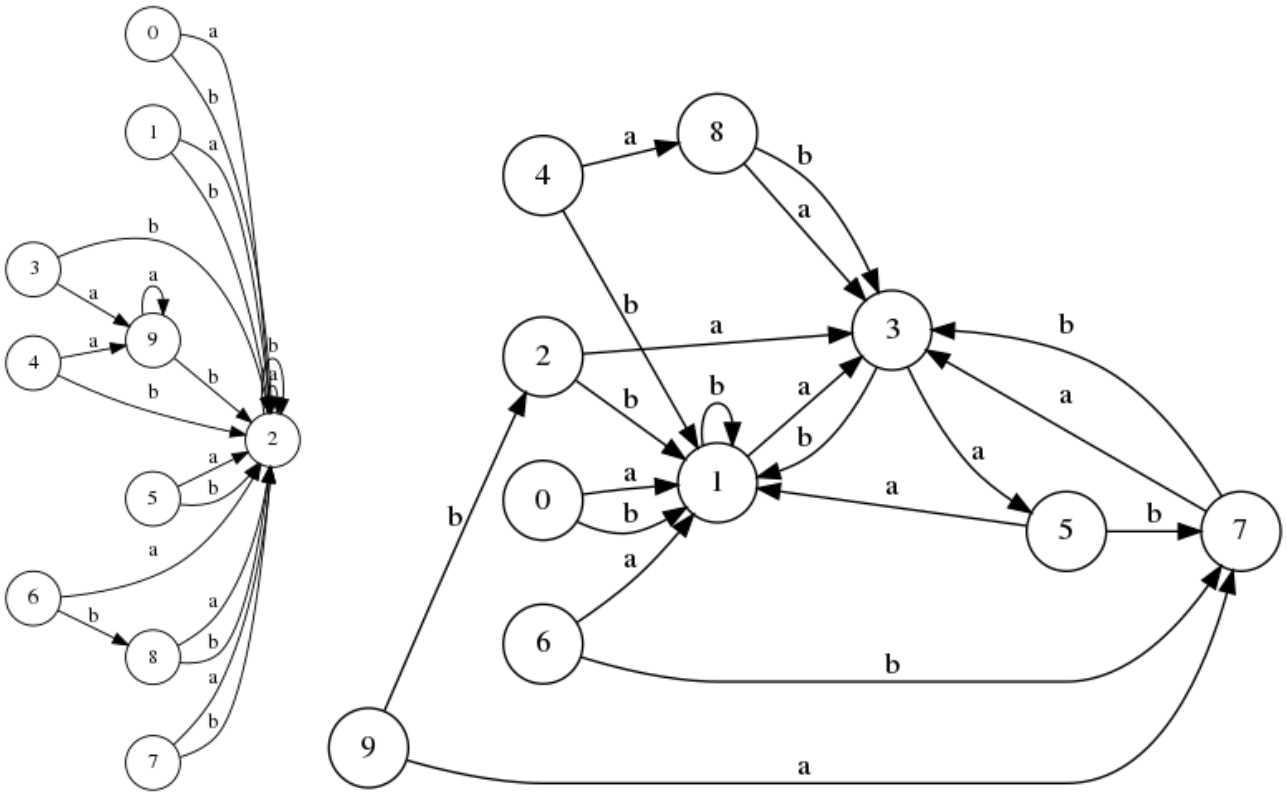
Algorithmes de recherche de séquences



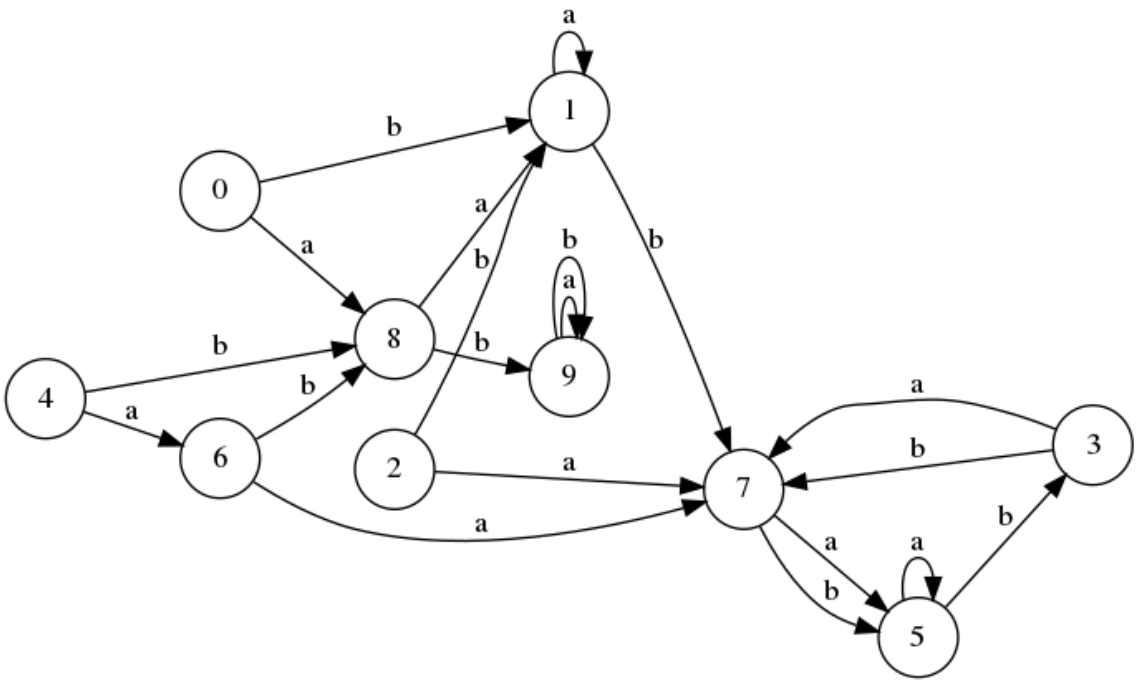
Algorithmes de tris



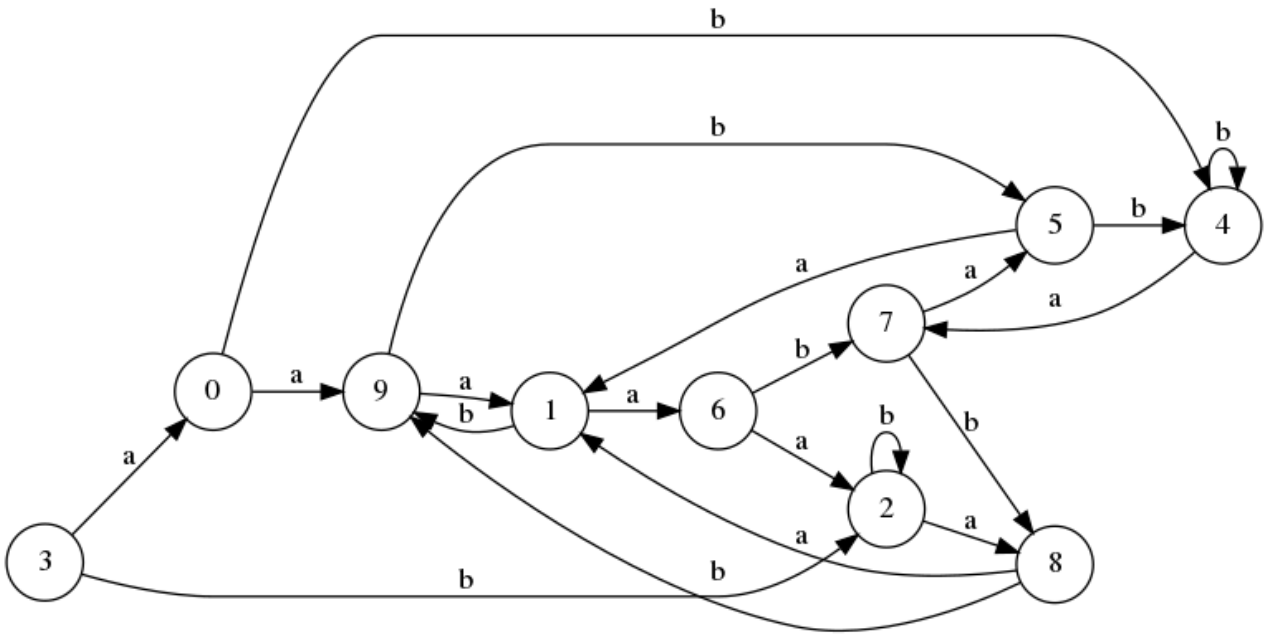
Automate aléatoire à entropie fixée $t = 2$ et $t = 3$



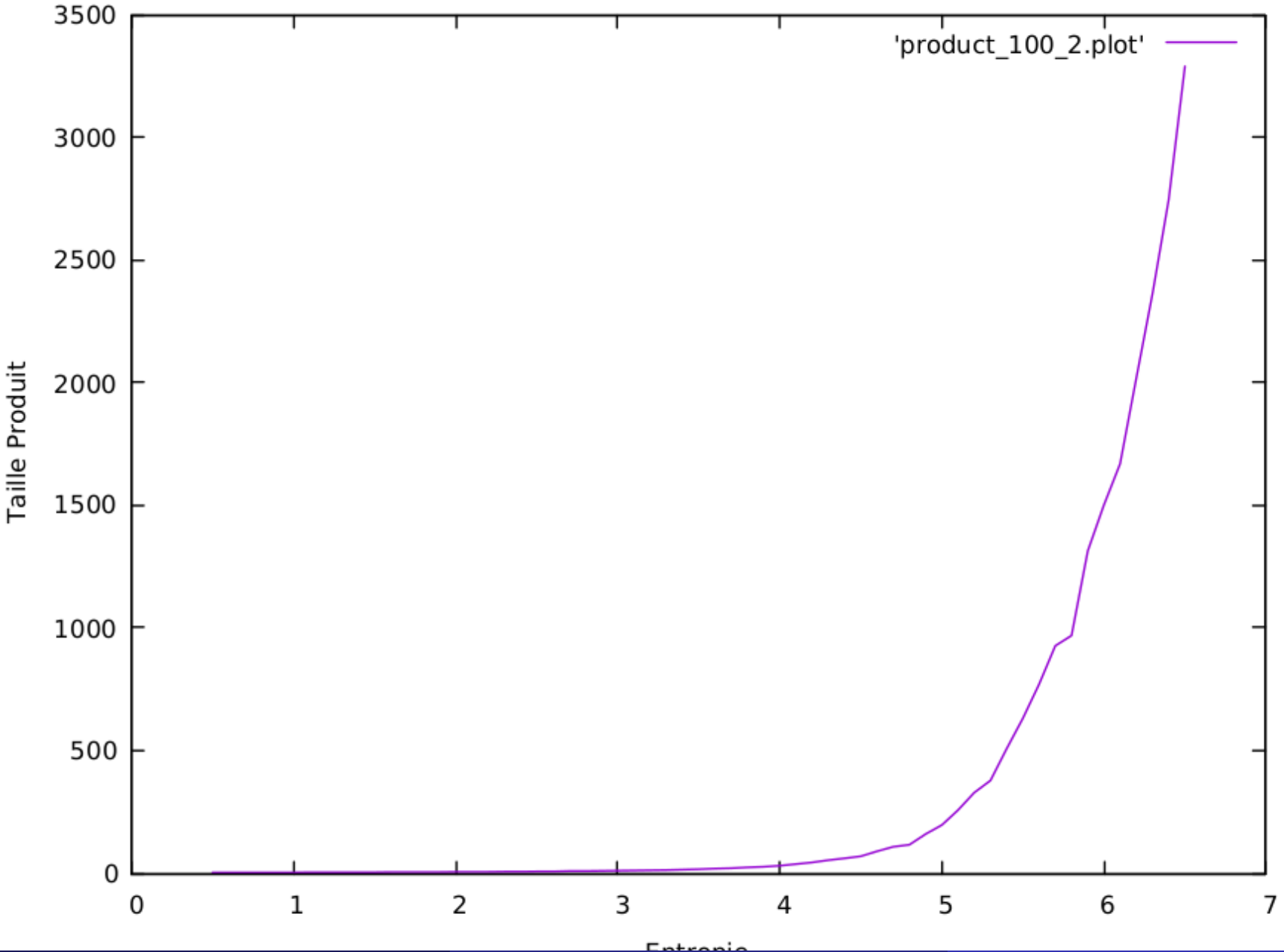
Automate aléatoire à entropie fixée $t = 4$



Automate aléatoire à entropie fixée $t = 5$



Produit d'automates



Merci !