Problème de l'alignement multiple de séquences avec SP-Score

M2 Graphes, Complexité & Combinatoire

Victor DEGUISE, Basile SUGRANES & François VICTOR

Résolution d'un problème avec un solveur de contrainte

Table des matières

Contexte

- recherche de régions hautement conservées parmi un ensemble de séquences biologiques
- déduction de l'histoire évolutive de certaines espèces
- La version de décision pour trouver l'alignement optimal avec SP-score a été prouvée **NP-complete** par Wang et al. en 1994

  –> La réduction provient du problème de la plus courte séquence commune.

Multiple Sequence Alignment

- Une séquence s est une chaîne de caractères sur un certain alphabet Σ (ici les 20 amino-acides)
- $S = \{s_1, s_2, ..., s_k\}$ soit un ensemble de k séquences
- A est l'alignement de $S' = \{s'_1, s'_2, ..., s'_k\}$ qui sont tous de la même longueur et qui maximisent le score de qualité.

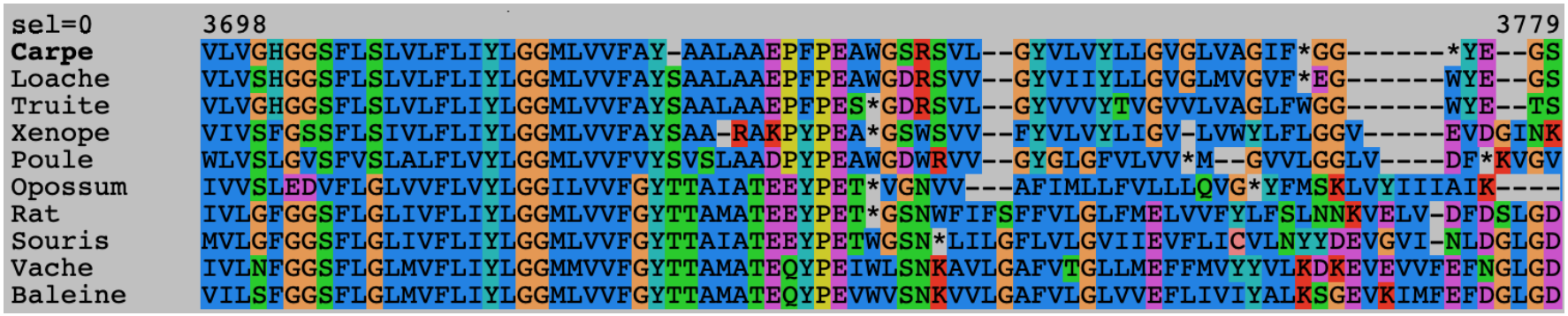


Figure 1 – example d'un alignement multiple de séquences

Approche Historique

- Programmation dynamique, Needleman & Wunsch, 1970.
  –> Première approche exacte, peu efficace $O(n^k)$

- Programmation linéaire, Althaus et al, 2006.
  –> ensemble d'alignement par paire représenté sous forme de graphe

- Approche de relaxation convexe utilisant la norme atomique, Yen et al, 2016.

Plus longue sous-séquence commune

Contrainte 1 : Des symboles $s_i[p]$ et $s_j[q]$ ne peuvent faire partie du longest common subsequence ssi ils sont identiques :
$x_{i,j,p,q} \leq m_{i,j,p,q}$

Contrainte 2 : Un symbole $s_i[p]$ peut être aligné avec au plus un symbole $s_j[q]$

Contrainte 3 : Un symbole $s_j[q]$ peut être aligné avec au plus un symbole $s_i[p]$

Contrainte 4 : Un symbole $s_i[p]$ qui est aligné avec un symbole $s_j[q]$ doit aussi être aligné avec un autre symbole $s_k[r]$ (Propagation)

Contrainte 5 : Les symboles qui sont solutions doivent appaître dans le même ordre dans toutes les séquences

Basé sur la plus longue sous-séquence commune

Contrainte 1 : Relachée

Contrainte 2 : Un symbole $s_i[p]$ peut être aligné avec au plus un symbole $s_j[q]$

Contrainte 3 : Un symbole $s_j[q]$ peut être aligné avec au plus un symbole $s_i[p]$

Contrainte 4 : Un symbole $s_i[p]$ qui est aligné avec un symbole $s_j[q]$ doit aussi être aligné avec un autre symbole $s_k[r]$ (Propagation)

Contrainte 5 : Les symboles qui sont solutions doivent appaître dans le même ordre dans toutes les séquences

Prise en compte de la matrice de substitution et ajout de pénalité de gap

Matrice de substitution et pénalité de gap

	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
A	0	1	0	4																	A
G	-3	0	-2	0	6																G
P	-3	-1	-1	-1	-2	7															P
D	-3	0	-1	-2	-1	-1	6														D
E	-4	0	-1	-1	-2	-1	2	5													E
Q	-3	0	-1	-1	-2	-1	0	2	5												Q
N	-3	1	0	-2	0	-2	1	0	0	6											N
H	-3	-1	-2	-2	-2	-2	-1	0	0	1	8										H
R	-3	-1	-1	-1	-2	-2	-2	0	1	0	0	5									R
K	-3	0	-1	-1	-2	-1	-1	1	1	0	-1	2	5								K
M	-1	-1	-1	-1	-3	-2	-3	-2	0	-2	-2	-1	-1	5							M
I	-1	-2	-1	-1	-4	-3	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-1	-4	-3	-4	-3	-2	-3	-3	-2	-2	2	2	4					L
V	-1	-2	0	0	-3	-2	-3	-2	-2	-3	-3	-3	-2	1	3	1	4				V
W	-2	-3	-2	-3	-2	-4	-4	-3	-2	-4	-2	-3	-3	-1	-3	-2	-3	11			W
Y	-2	-2	-2	-2	-3	-3	-3	-2	-1	-2	2	-2	-2	-1	-1	-1	-1	2	7		Y
F	-2	-2	-2	-2	-3	-4	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	1	3	6	F
	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	

Choix du solver

OR-Tools

linear programming

```
# Create the linear solver with the GLOP backend.
solver = pywraplp.Solver.CreateSolver('GLOP')
if not solver:
    return
```

Mixed integer programming

```
# Create the mip solver with the SCIP backend.
solver = pywraplp.Solver.CreateSolver('SCIP')
if not solver:
    return
```

Benchmark

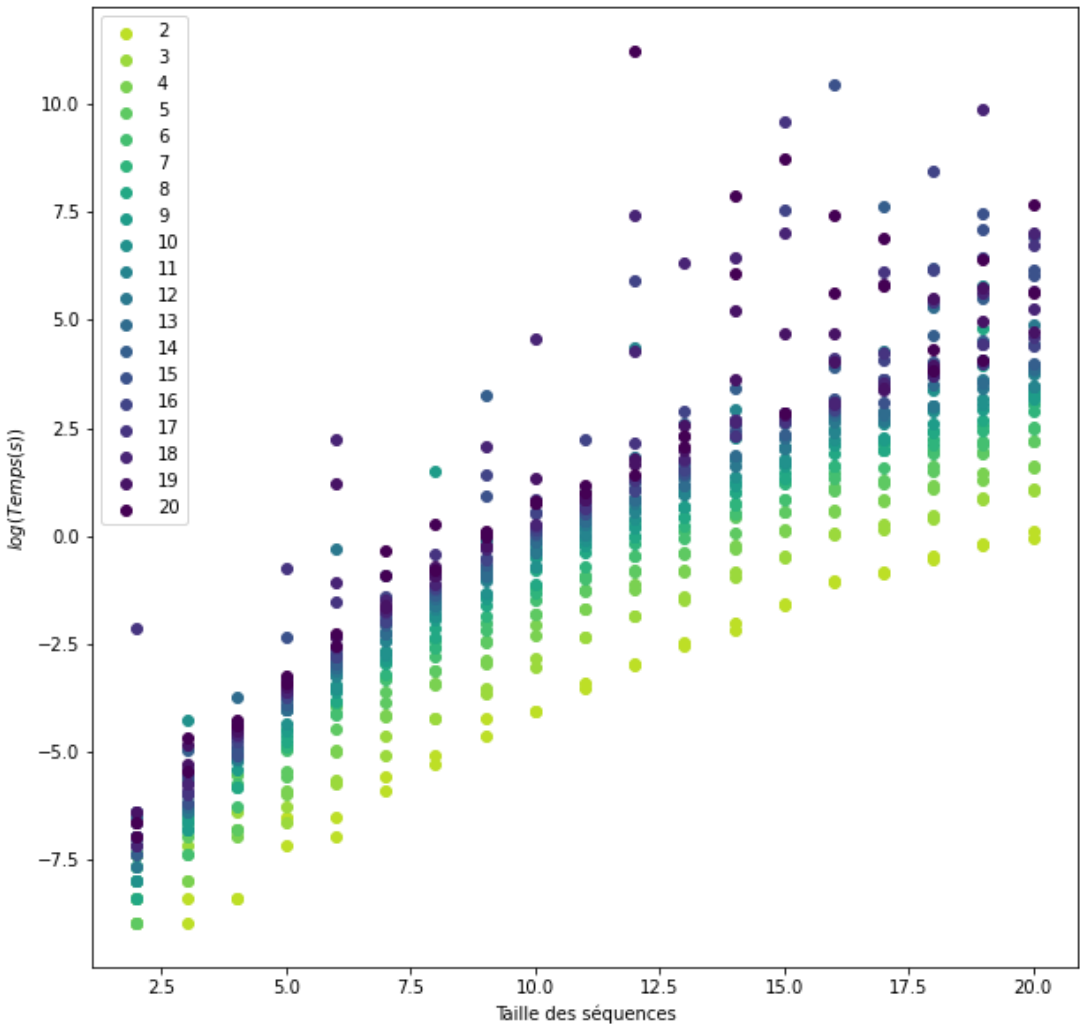
Générateur de séquence

- modèle substitution-délétion
- instances du problème : taille des séquences de 2 à 20 et ensemble de séquences de 2 à 20

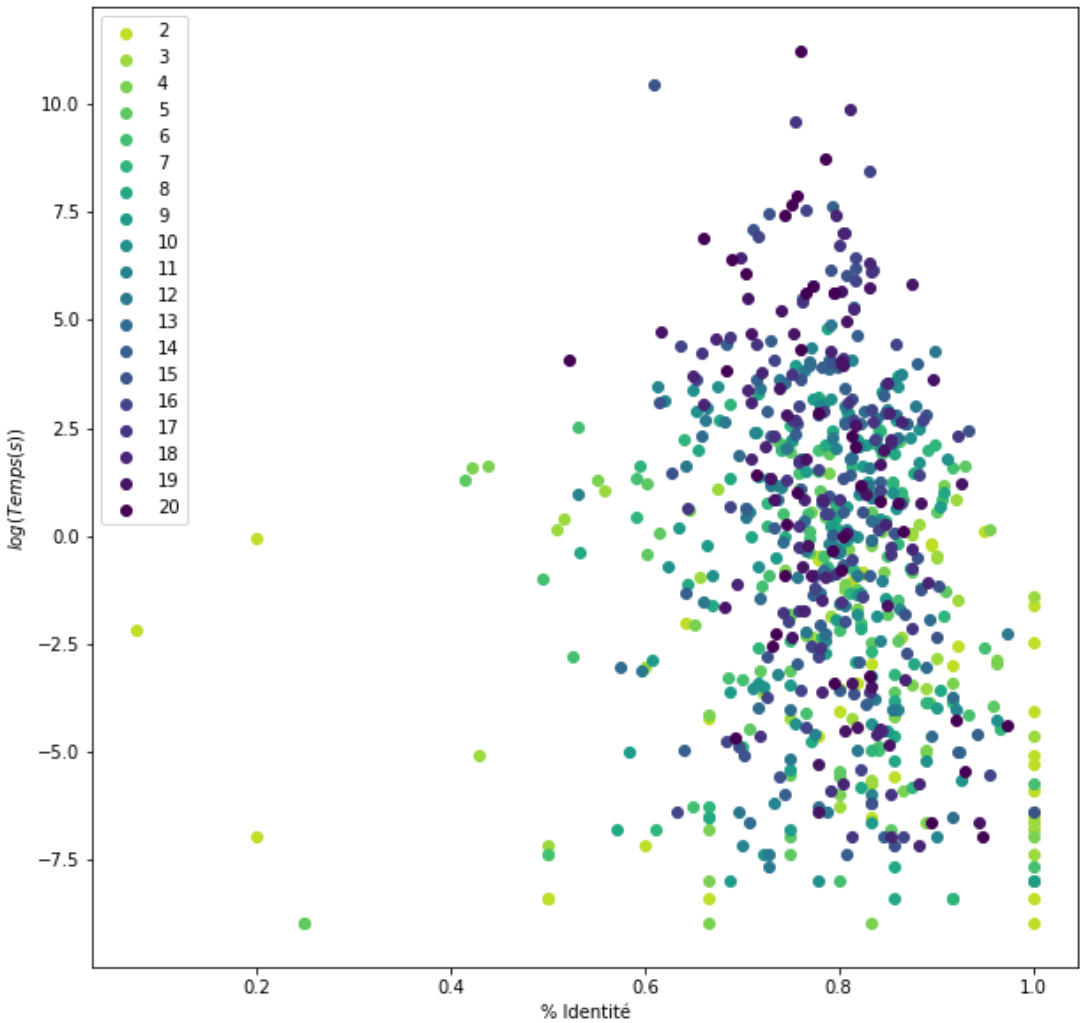
```
>seq1
HAHVEAPRIMEKYLQKMSVK
>seq2
HAHVEAPRIMEKYLQKMSVK
>seq3
HAIVEAPRNMKYLQKMSVK
>seq4
HAHVEAPRIMEKYLQKMSVK
>seq5
HAHQEAPRIMEKYLQRMSVK
```

```
>seq6
HAHVEAPRIMEKYLQKMSWK
>seq7
HAVVEAPRIMVKYWYKMSDK
>seq8
HAHVEAPRIMEKYLQKMSVK
>seq9
HAHVEAPLAMEKYLQKYSVK
>seq10
HAHVEAPRTMEKYLQKMSVK
```

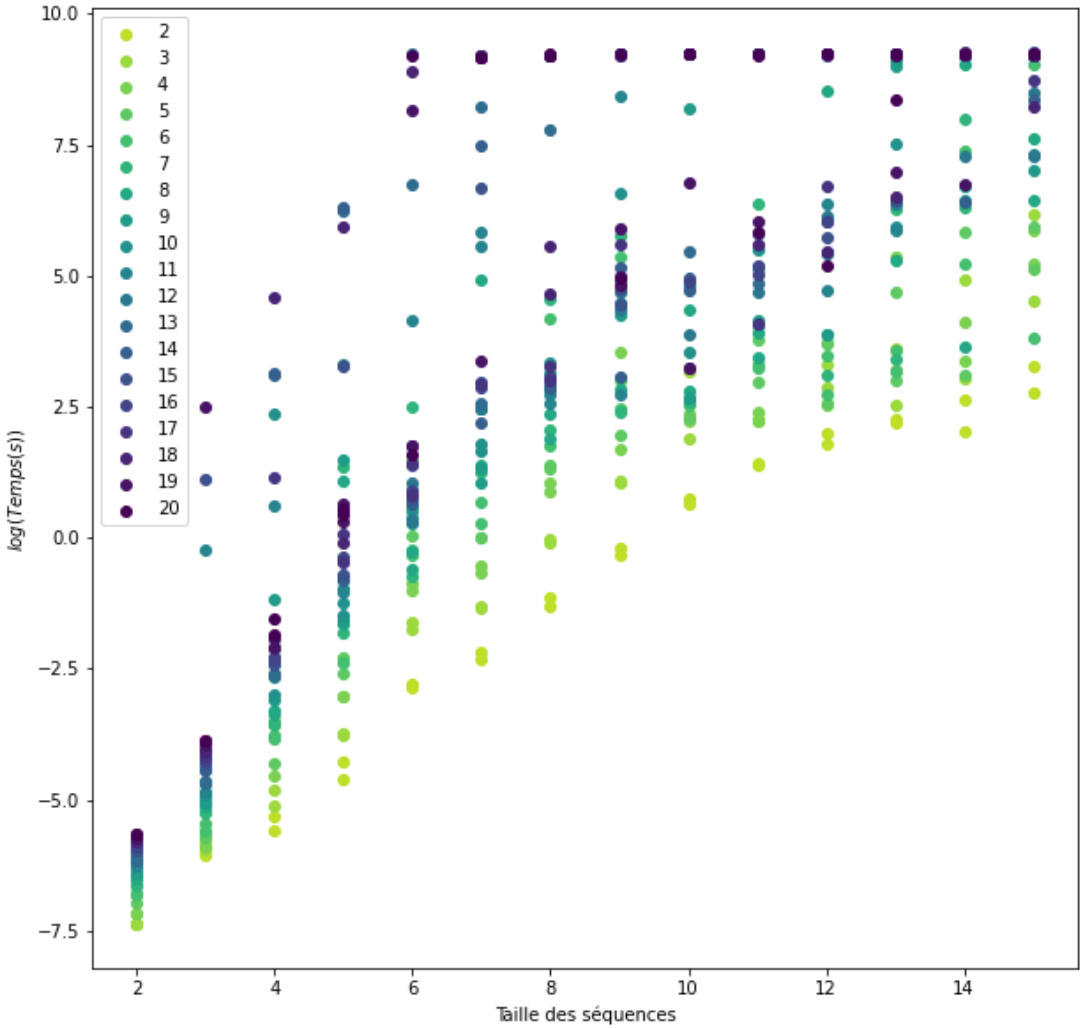
LCS : *Log*(*temps*) en fonction de la taille et du nombre de séquences



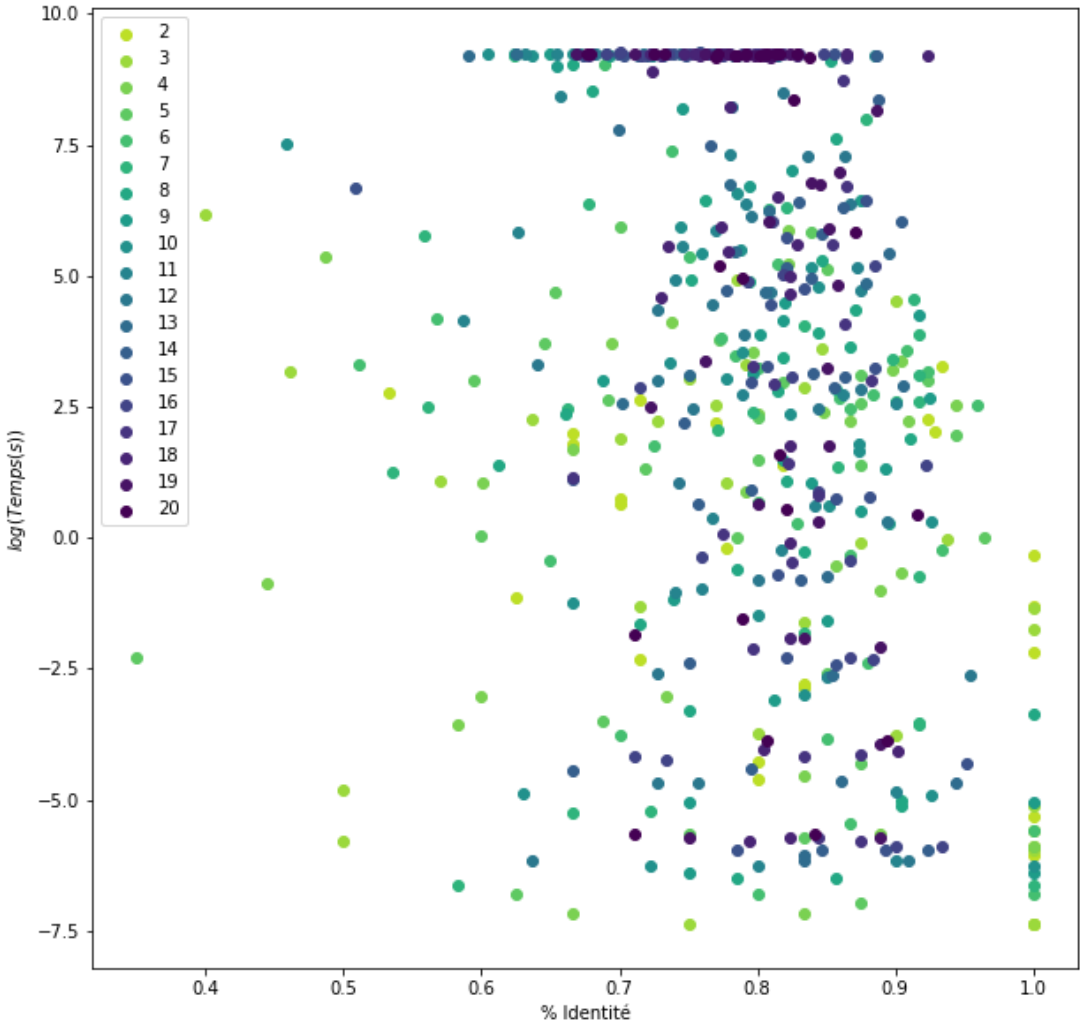
LCS : *Log*(*temps*) en fonction du pourcentage d'identité des séquences



MSA : *Log*(*temps*) en fonction de la taille et du nombre de séquences



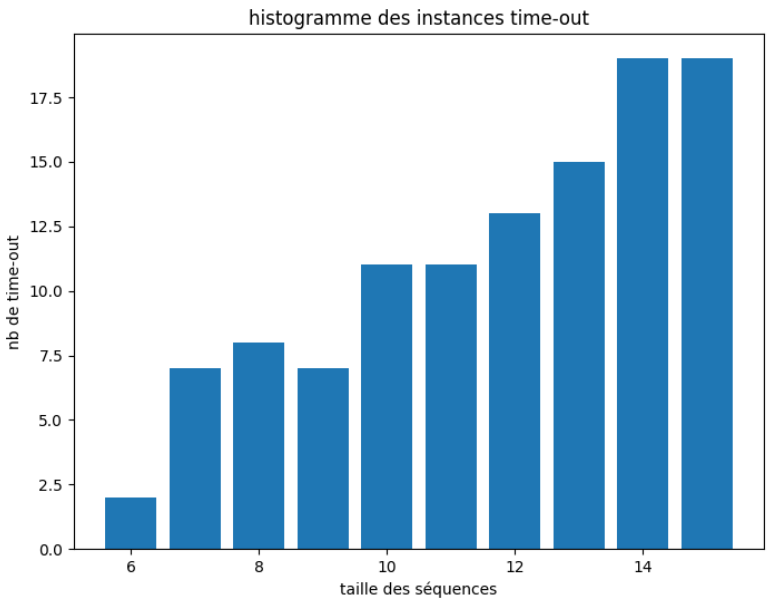
MSA : *Log*(*temps*) en fonction du pourcentage d'identité des séquences

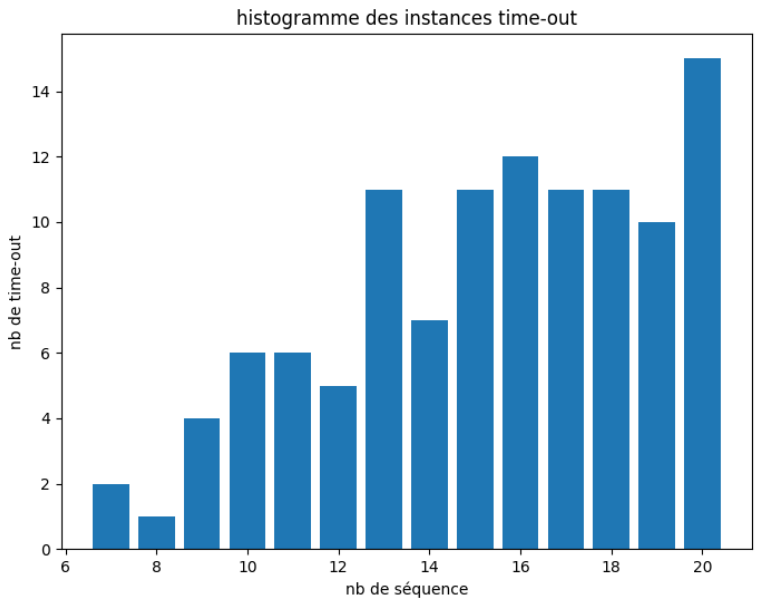


Instances time-out

taille_seq	6	7	8	9	10	11	12	13	14	15	
nb_timeout	2	7	8	7	11	11	13	15	19	19	Tot : 112

nb_seq	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
nb_timeout	2	1	4	6	6	5	11	7	11	12	11	11	10	15	Tot : 112





Conclusion & limitations

- Limites d'implémentation dans OR-Tools
  –> linéarisation de contrainte par Big-M méthode

- Modèle implémenté trop gourmand en ressources en théorie