Algorithmes de filtrage

Nadia El-Mabrouk

Plan

1. Introduction
   - Objectif
   - Qu'est-ce qu'une heuristique?
2. Algorithmes de filtrage: Principe et méthode exacte (Baeza-Yates-Perlberg, 1992)
3. Heuristique FASTA
4. Optimisation possible en introduisant le voisinage d'un k-mer
5. Heuristique BLAST
6. Graines espassées. Méthode de PatternHunter

1. Introduction

Recherche de *P* de taille *m* dans *T* de taille *n* à *d* erreurs près

Programmation dynamique: Temps O(mn)

Différentes améliorations: Temps O(dn)

Est-ce qu'on peut faire mieux? Temps sous-linéaire?

Heuristique ou méthode exacte?

- Heuristique pour la recherche d'un motif: méthode permettant de trouver la « plupart » des occurrences, mais pas de garantie de les trouver toutes, et peut se tromper.
  - Faux-négatifs: Occurrences non détectées.
  - Faux-positifs: Motifs trouvés qui ne sont pas des occurrences.

Heuristique

- Sélectivité (spécificité): Capacité à ne détecter que la réalité biologique et rien de plus.

  Problème des Faux-Positifs

- Sensibilité: Capacité à détecter tout ce qui est intéressant sur le plan biologique.

  Problème des Faux-Négatifs

2. Algorithmes de filtrage

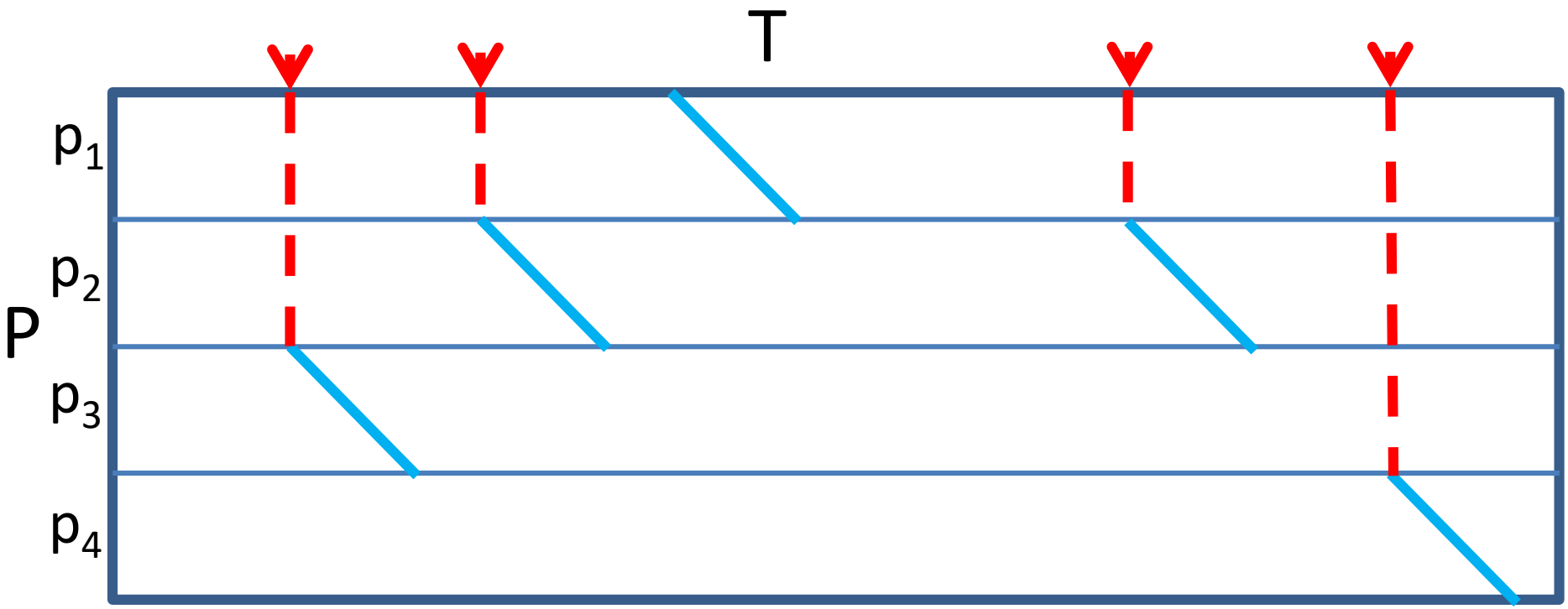
Effectuer un premier passage sur *T* pour éliminer toutes les parties qui ne sont pas susceptibles de contenir *P.*

- Partitionner *P* (ou *T*) en facteurs de taille *k* (k-mers, seeds ou graines). Construire un index de ces facteurs.
- Phase de recherche: Utiliser une méthode de recherche exacte pour trouver toutes les occurrences de ces facteurs dans *T*, en temps (sous) linéaire
- Phase de vérification: Utiliser une méthode de recherche approchée dans un intervalle restreint autour de chaque facteur trouvé, en temps (sous) linéaire.

Algorithmes de filtrage

- Les k-mers définissent des ancrages dans la table de programmation dynamique

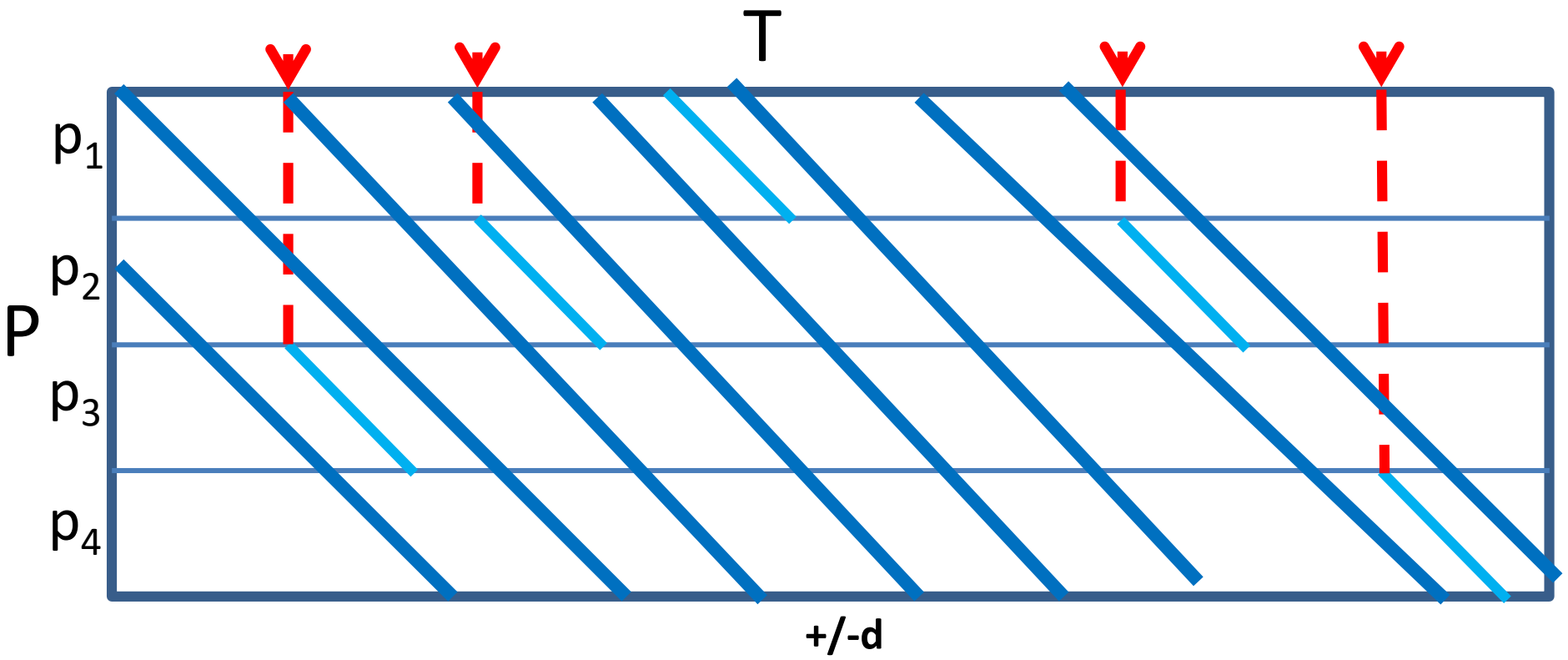
$P = p_1 p_2 p_3 p_4$



Algorithmes de filtrage

- Les k-mers définissent des ancrages dans la table de programmation dynamique

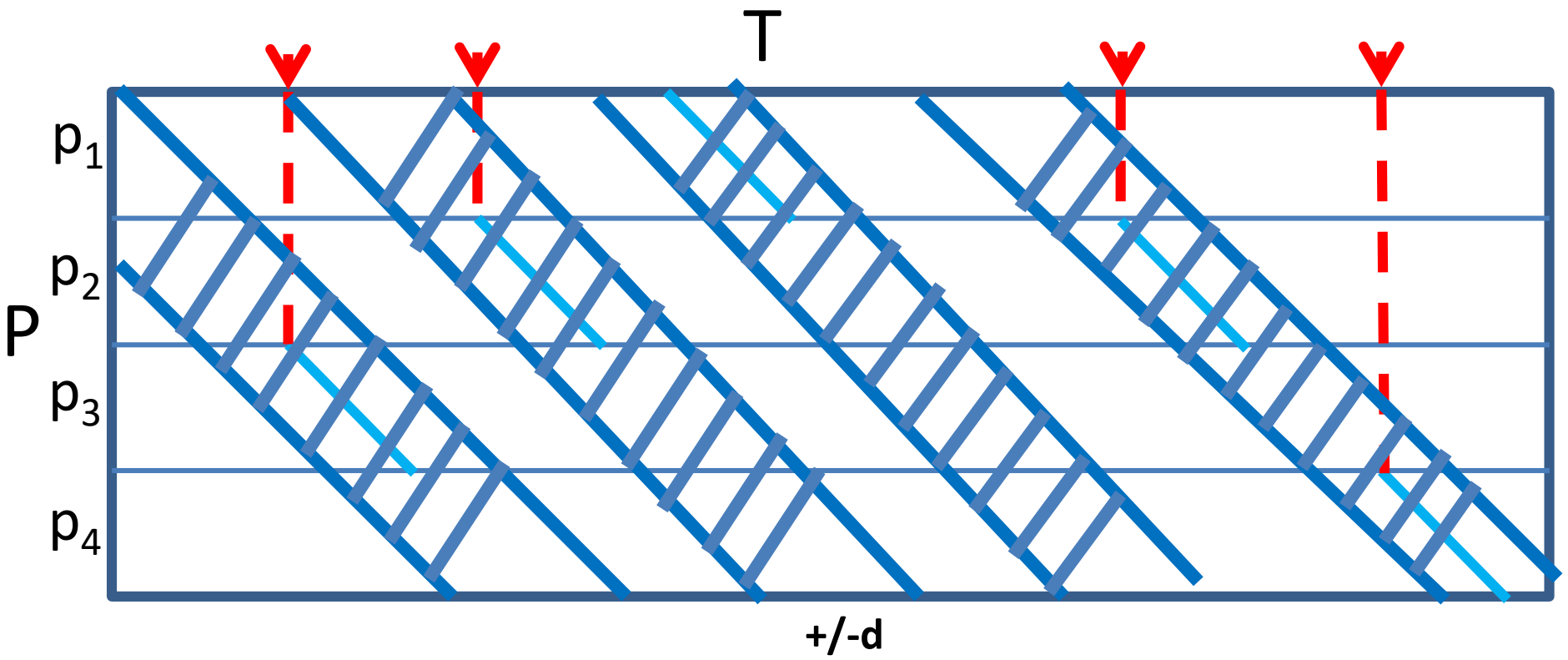
$P = p_1 p_2 p_3 p_4$



Algorithmes de filtrage

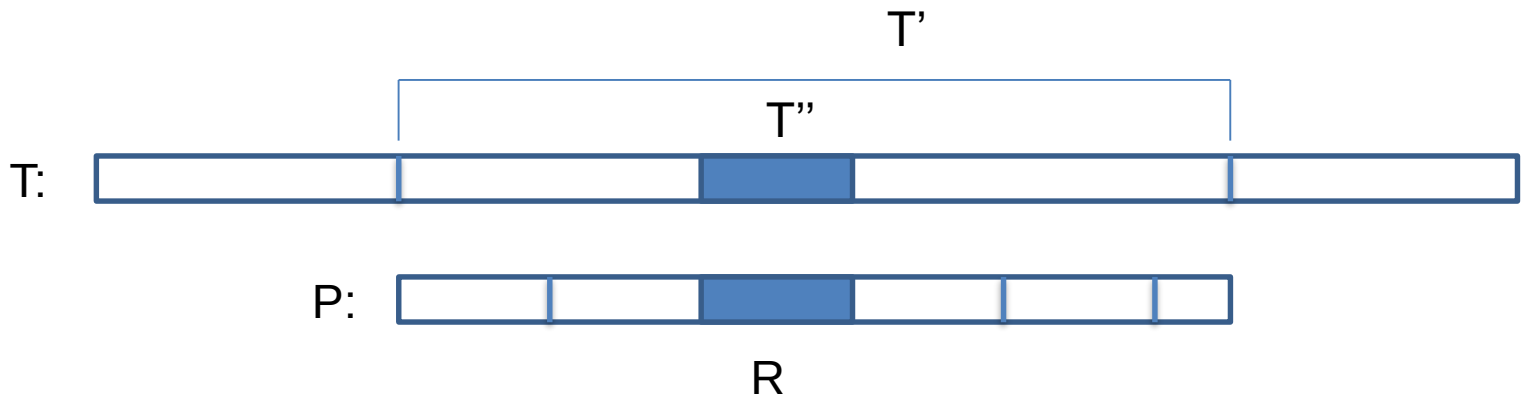
- Les k-mers définissent des ancrages dans la table de programmation dynamique

$P = p_1 p_2 p_3 p_4$



Une méthode exacte (Baeza-Yates-Perlberg, 1992)

- Partition de *P* en régions de taille *k = ENT(m/d+1)* → *d+1* régions de taille *k*, plus au plus une région de taille *< k.*
- Si le facteur *T'* de *T* est une occurrence de *P* à *d* erreurs près, alors il existe au moins une région *R* de *P* et un facteur de même taille *T''* de *T'* tel que *R* et *T''* coïncident exactement.



Idée générale (Baeza-Yates-Perlberg, 1992)

$\mathcal{P}$: Ensemble des *d+1* premières régions de *P*.

1. Construire un « index » de $\mathcal{P}$
2. Trouver l'ensemble $\mathcal{I}$ des pos. des occurrences de $\mathcal{P}$ dans *T*.
3. Étendre les occurrences par programmation dynamique.

- Construction d'un index possible en temps et espace *O(m)*: Arbre des préfixes (Aho-Corasick), « trie structure».
- Recherche exacte possible en *O(n)*
- Phase de vérification: *O(hdm)* où $h=|\mathcal{I}|$.

➔ O(m+n+hdm)

3. Heuristique FASTA (Lipman, Pearson 1985)

- Taille de graine *k* fixée (en général 6 pour nuc., 2 pour AA)
- Indexer tous les facteurs de taille *k* de *P (O(m))*
- Rechercher toutes les occurrences exactes de ces facteurs de *P* dans *T (O(m+n))*
- Chainer les graines pour former des alignements complets

~ temps proportionnel au nombre *h* de graines

Heuristique FASTA (Lipman, Pearson 1985)

1. Pour une valeur *ktup* donnée (en général 6 pour nuc.2 pour AA), trouver toutes les paires de séquences de taille *ktup* identiques dans P et T: hot-spot

2. Déterminer des zones denses en identité: hot-spot consécutifs sur chaque diagonale. Score d'une zone:
   1. Score positif pour chaque hot-spot
   2. Score négatif pour les espaces entre les hot-spot

   FASTA garde les 10 zones de score optimal. Zones contenant des matchs et mismatchs

3. Réaligner chaque zone, en considérant une matrice de substitution (PAM ou BLOSUM)

   Init1: Meilleur alignement obtenu

4. Parmi les 10 zones, garder celles dont le score dépasse un seuil ``cut-off''. Combiner les zones en une seule

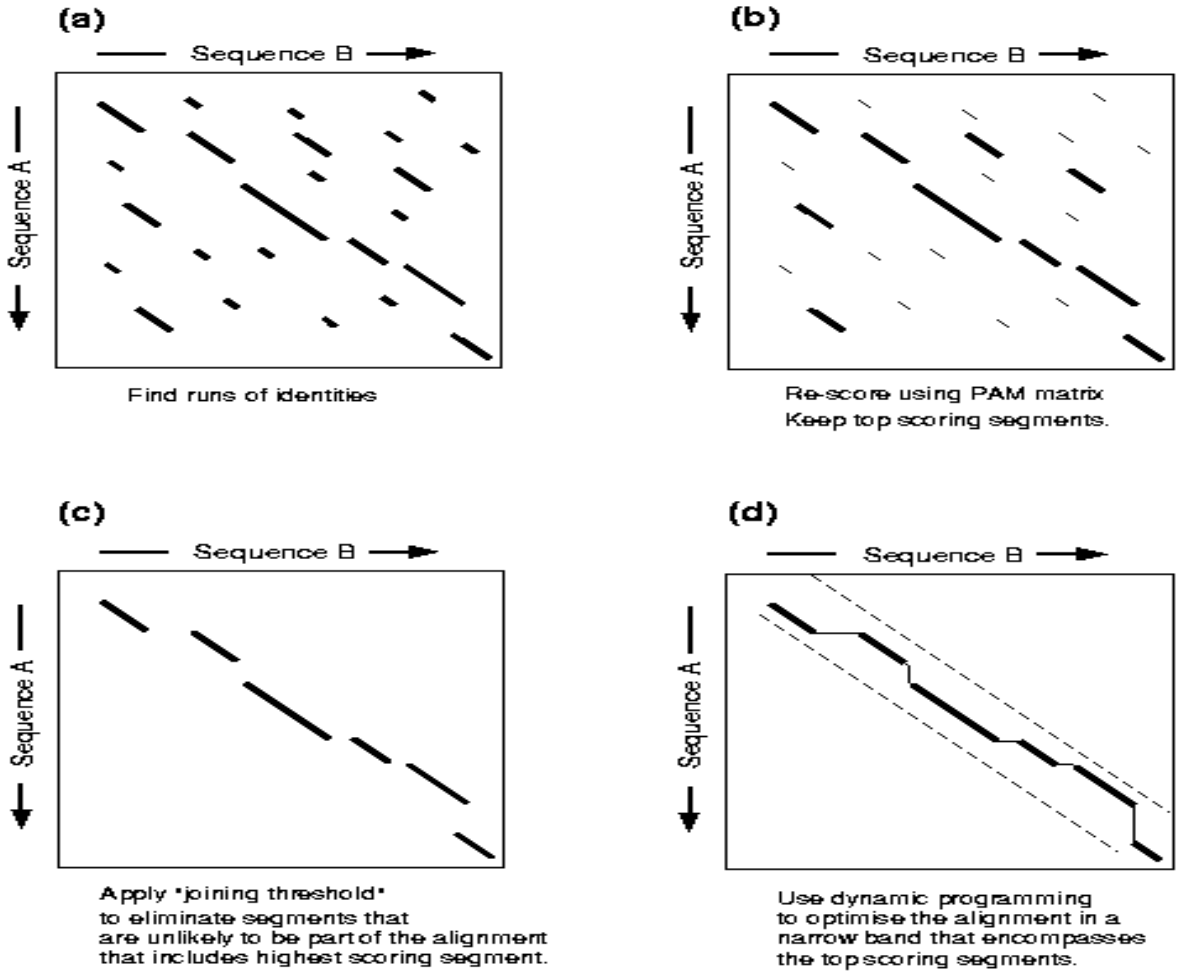
   Initn: Contient insertions/suppressions/mismatchs

5. Programmation dynamique dans une bande autour de *Init1* (bande de taille 16 si ktup=2)

   Opt: Meilleur alignement obtenu

Au cours de la recherche, statistiques calculées pour *Init1, Initn, Opt*: alignements significatifs ou non.

FASTA Algorithm



Taille des graines?

- Complexité de l'heuristique de filtrage dominée par *h*: nombre de graines.
- Combien de graines attendues pour deux séquences aléatoires sur un alphabet Σ*?*
  - Un mot de taille *k* a une probabilité de $1/|\Sigma|^k$ d'apparition dans l'une ou l'autre des séquences.

  ➔h de l'ordre de $nm\ |\Sigma|^{-k}$ ($4^{-k}nm$ dans le cas de séquences d'ADN)

  ➔Complexité totale de l'heuristique de filtrage $O(n+m+\ |\Sigma|^{-k}ndm^2)$

- D'autant plus efficace que *k* est grand
- Mais plus *k* est grand plus on a de chances de manquer des occurrences.

4. Voisinage d'un mot

- Plutôt que de rechercher les occurrences exactes des k-mers, rechercher un « voisinage » des k-mers: mots qui sont à moins de $\varepsilon\%$ d'erreurs, ou à une distance $\leq d$ $(d = \varepsilon\, m)$

$N_d(w) = \{ v \,/\, v$ et w ont au plus d différences$\}$

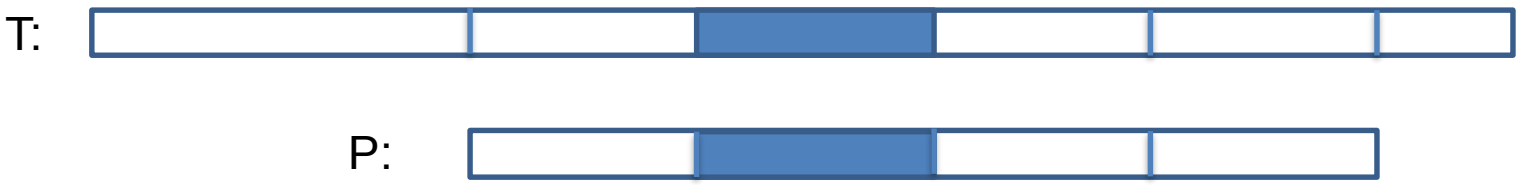
$N_1(abbaa) = \{$ aabaa, aabbaa, abaa, abaaa, ababaa, abba, abbaa, abbab, abbaaa, abbaba, abbba, abbbaa, babbaa, bbaa, bbbaa$\}$

(1-match ou 20%-match)

Puissance du voisinage

- Supposons que l'on cherche un 3-match d'un mot *P* de taille 40.
- Si on choisit *k=10* et qu'on divise *P* en 4 10-mers, alors au moins un doit matcher exactement.

=> Occurrence tous les $|\Sigma|^{10}/4$ caractères (e.g. $2.5 \cdot 10^5$ pour les nucléotides)



Puissance du voisinage

- Supposons que l'on cherche un 3-match d'un mot *P* de taille 40.
- Si on choisit *k=20* et qu'on divise *P* en 2 20-mers, alors au moins un 1-voisin des deux doit matcher.

=> Occurrence tous les $|\Sigma|^{20}/2|N_1(20)|$ caractères (e.g. $10^{12} / 2 \cdot 160 = 3.12 \cdot 10^9$ pour les nuc.)

10,000 fois plus spécifique ! (mais beaucoup plus de mots pour la recherche exacte)

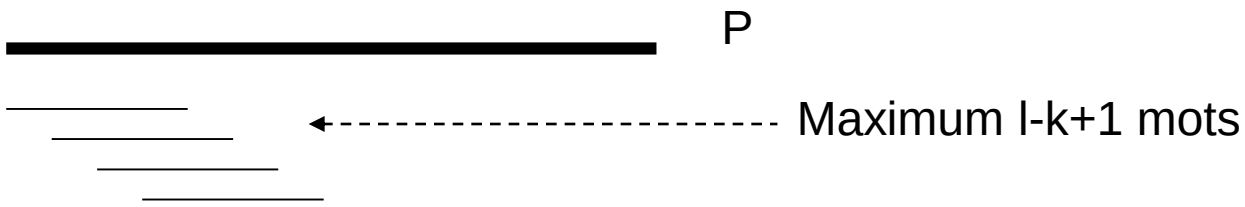
5. BLAST

- Recherche exacte sur un voisinage des k-mers
- Distance Hamming pondérée (e.g. PAM120)
- Extension: stop quand le score chute sous une valeur seuil.
- BLAST est une heuristique

"BLAST" = Basic Local Alignment Search Tool

BLAST

- Former la liste de tous les k-mers (seeds ou graines) de la séquence requête *P*



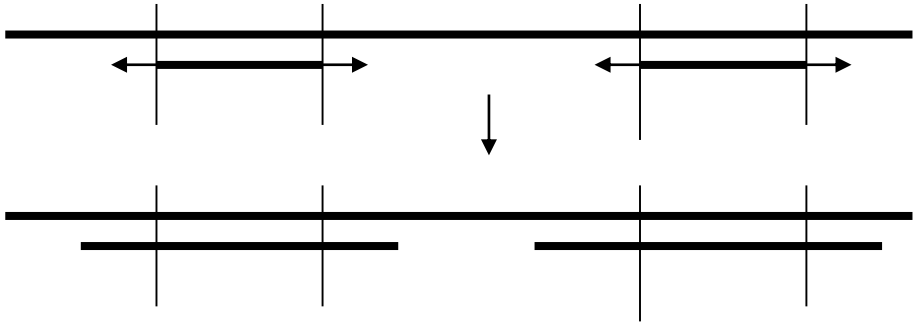
- Pour chaque facteur *w*, former la liste de tous les mots de taille *k* dont le score avec *w* dépasse un seuil *s*

Exemple: Pour w =PQG, {PQG, PRG, PKG, PDG, PMG…}

- Identifier les occurrences exactes des mots de la liste dans la BD

High Scoring Pairs (HSP)

- Pour chaque paire de séquences trouvées, étendre l'alignement dans les deux directions, jusqu'à ce que le score de l'alignement chute de *X* par rapport à sa valeur d'origine. Segment accepté si score > *S*



- Le HSP de score maximal sur l'ensemble de la séquence est appelé Maximal Scoring segment Pair (MSP)
- Les alignements locaux HSP sont chaînés pour former des alignements plus longs, incluant des espaces et des trous.

Si le MSP ou les HSP combinés ont un score qui dépasse un certain seuil *S,* il sont affichés

Paramètres

- La séquence format FASTA
- La banque (compressée)
- k (taille du mot).
  - Protéines: k de 3 à 5, et s = 17

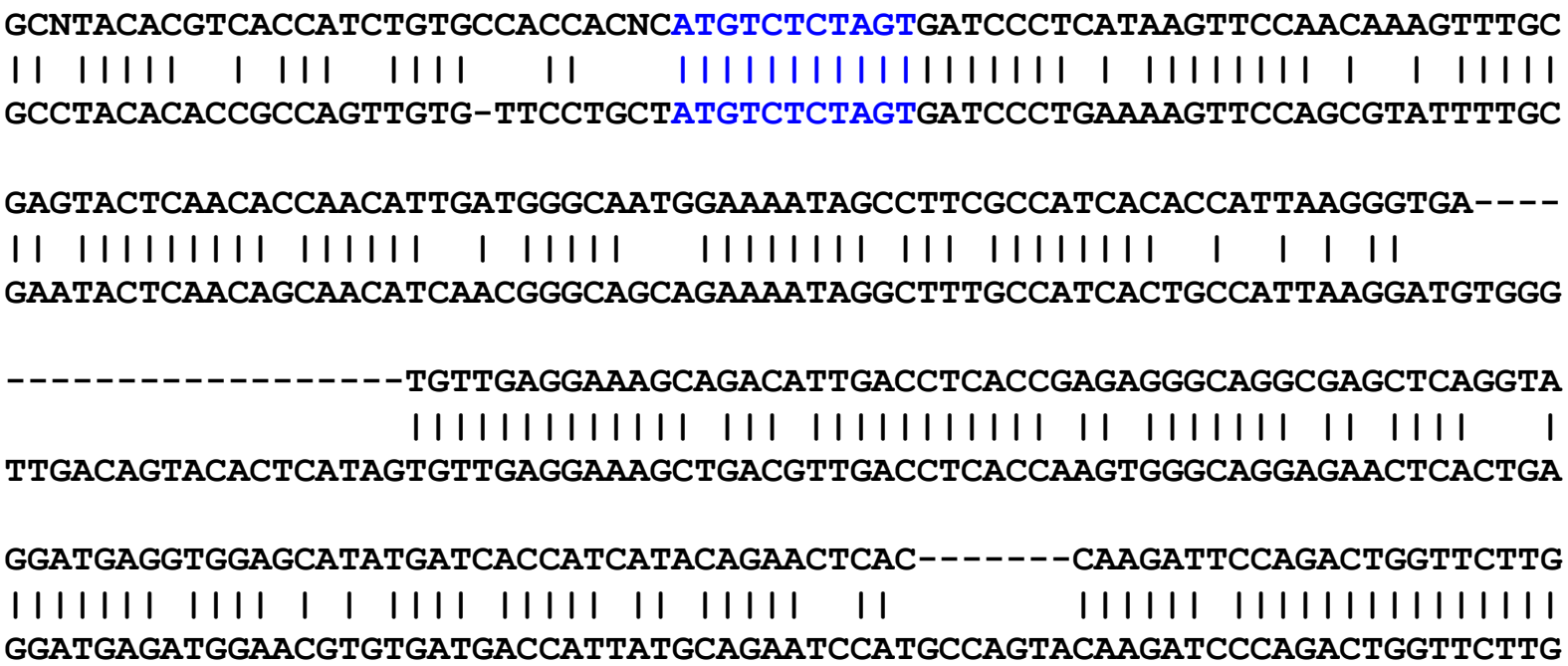
  Donne à peu près 50 mots pour chaque facteur
  - Nucléotides: k = 11 ou 12
- S (seuil de sélection d'un score)
- Matrices de substitution (BLOSUM 62) ou score pour les nucléotides (+5/-4)

Évaluation statistique

- Expect-value = nb de fois où un HSP est attendu par chance sur l'ensemble de la banque. Plus cette valeur est faible, plus le HSP est significatif
- P-value: P(N): Probabilité du score observé. Plus cette valeur est faible, plus le HSP est significatif.

7. Graines espacées

BLAST trouve une graine de taille 11 qui match, puis étend



Exemple d'une occurrence manquée (Exemple de B. Ma)

- Pas de graine de taille 11 qui match, pourtant similarité de 80%:

```
GAGTACTCAACACCAACATTAGTGGGCAATGGAAAAT
||  |||||||||  ||||||  |  ||||||    ||||||
GAATACTCAACAGCAACATCAATGGGCAGCAGAAAAT
```

- Dilemme:
  - Sensibilité – nécessite des graines courtes
    - Capacité à détecter les homologies
  - Rapidité – nécessite des graines plus longues
    - Mega-BLAST utilise des graines de taille 28.

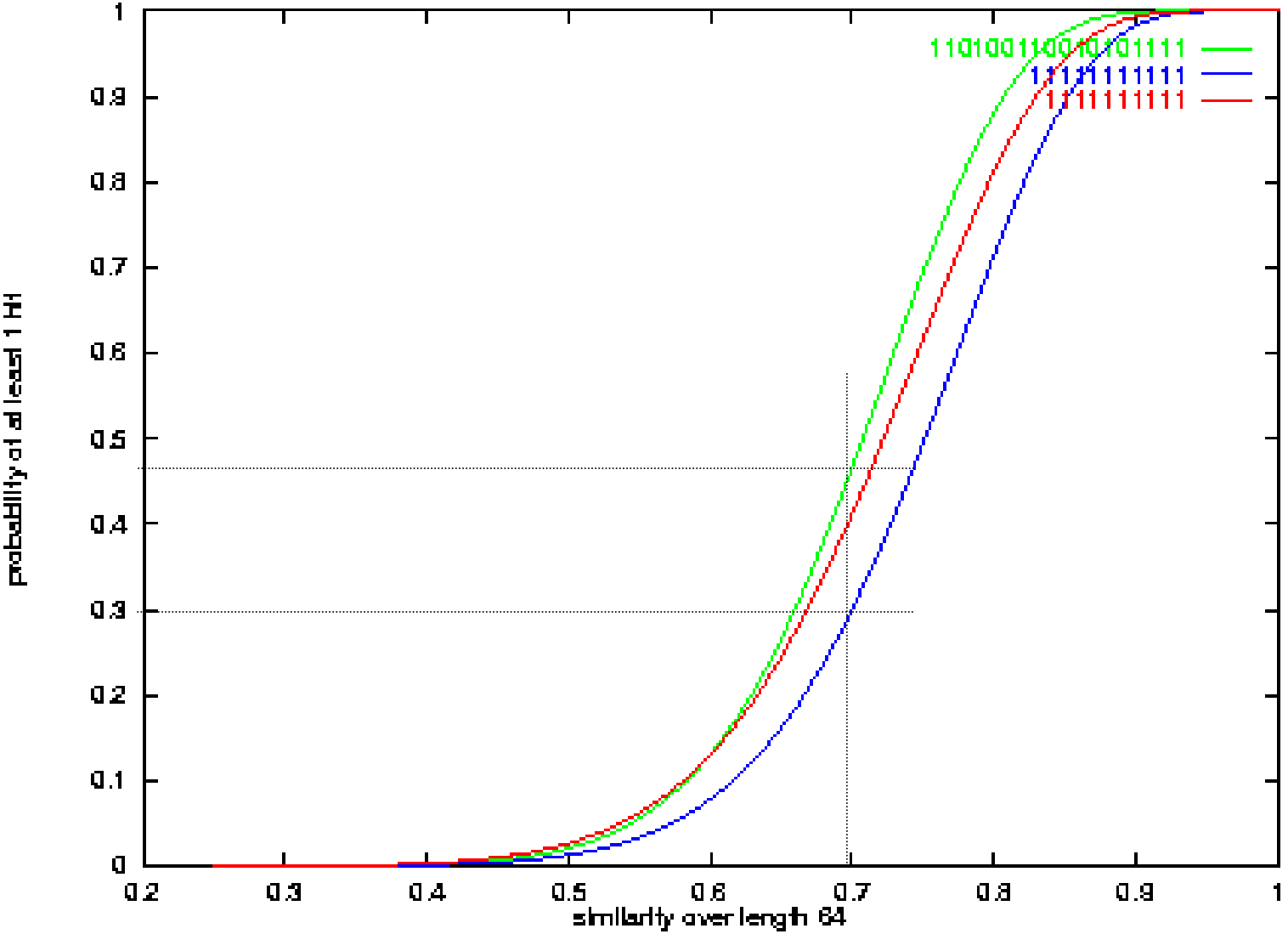
PatternHunter utilise des “graines espacées”

- 111010010100110111 (appelé modèle)
  - 11 matchs requis (poids=11)
  - 7 positions “don’t care”

```
GAGTACTCAACACCAACATTAGTGGCAATGGAAAAT...
||  |||||||||  |||||  ||  |||||    ||||||
GAATACTCAACAGCAACACTAATGGCAGCAGAAAAT...
        1110100101001101111
```

- Hit = Tous les matchs requis sont satisfaits
- Modèle de BLAST = 1111111111

VI. Simulated sensitivity curves



Pourquoi sensibilité meilleure?

- Les copies ‘shiftées’ des graines espacées ne chevauchent pas trop:

```
111010010100110111
 111010010100110111
  111010010100110111
   111010010100110111
    111010010100110111
     111010010100110111
      111010010100110111
        ......
```

```
11111111111
 11111111111
  11111111111
   11111111111
     ......
```

- Les ‘Hits’ à différentes positions sont plus indépendants
- Plus les copies shiftées sont indépendantes, plus on augmente la probabilité d’identifier une homologie. Moins il y a de similarités entre deux copies shiftées, plus le modèle est susceptible de donner une bonne sensibilité.

VI. Pourquoi plus rapide avec des graines espacées?

```
TTGACCTCACC?
|||||||||||?
TTGACCTCACC?
11111111111
 11111111111
```

```
CAA?A??A?C??TA?TGG?
|||?|??|?|??||?|||?
CAA?A??A?C??TA?TGG?
11101001010011011 1
 11101001010011011 1
```

- Une homologie donne lieu à plusieurs ‘hits’ par BLAST (redondance)

- Graines espacées donnent lieu à moins de ‘hits’ pour chaque homologie

Observations (B. Ma)

- Dans une séquence cible de taille 64 qui contient 70% de similarité avec la séquence requête, et qui contient un hit de taille 11, la moyenne des hits dans cette région :
  - 2.0 pour la modèle PH (graine espacée)
  - 3.6 pour le modèle BLAST.

Observations (B. Ma)

- Des modèles différents peuvent détecter différentes homologies
- Deux conséquences:
  - Certains modèles sont meilleurs que d'autres
  - On peut utiliser simultanément plusieurs modèles de graines
    - Approcher les 100% de sensibilité.
    - PatternHunter II