Méthodes de *mapping* de *reads* avec indexation des *reads*

Pierre Morisse, Thierry Lecroq, Arnaud Lefebvre

LITIS
Université de Rouen
76000 Rouen, France

Introduction

Depuis le milieu des années 2000 et le développement des séquenceurs à très haut débit (*Next Generation Sequencing*), la biologie doit faire face au traitement d'énormes quantités de données, formées par des millions de très courtes séquences appelées *reads*. Dans un papier relativement récent, Philippe et al. ont souligné l'importance de l'indexation de ces *reads* afin de résoudre des problèmes de correction ou de $mapping$, et ont développé un index supportant les 7 requêtes suivantes, pour une séquence f de longueur k donnée :

- Dans quels *reads* f apparaît ?
- Dans combien de *reads* f apparaît ?
- Quelles sont les occurrences de f ?
- Quel est le nombre d'occurrences de f ?
- Dans quels *reads* f n'apparaît qu'une fois ?
- Dans combien de *reads* f n'apparaît qu'une fois ?
- Quelles sont les occurrences de f dans les *reads* où f n'apparaît qu'une fois ?

Nous présentons ici l'état de l'art concernant les technologies de séquençage et les problèmes susmentionnés, ainsi que le problème sur lequel nous nous sommes principalement penchés durant le déroulement de ce stage.

État de l'art

Les tableaux présentés ci-dessous résument brièvement l'état de l'art concernant les technologies de séquençage et les outils existants, utilisant une structure d'index sur les *reads*, et permettant de résoudre des problèmes de correction, de $mapping$, ou de traitement des 7 requêtes précédentes.

Technologie	Technique de séquençage	Plateforme	Nombre de $reads$	Longueur	Précision	Temps	Débit	Coût	Erreurs
Illumina	Synthèse basé sur polymères	HiSeq 2500/1500 MiSeq	3 milliards 17 millions	36 - 100 25 - 250	99 >99	2 - 11 jours 4 - 27 heures	600 8,5	740 000 125 000	Substitutions
Roche	Polyséquençage	454 GS FLX+ 454 GS Junior	1 million 1 million	700 400	99,997 >99	23 heures 10 heures	0,7 0,4	450 000 108 000	Indels.
ABI Life Technologies	Ligatures Détection de protons	5500xl SOLiD Ion Proton Chip I/II	2,8 millions 60 - 80 millions	75 jusqu'à 200	99,99 >99	7 jours 2 heures	180 10 - 100	595 000 243 000	Indels.
Pacific Biosciences	Simple molécule en temps réel	PacBio RS	50 000	3 000 en moyenne	85	2 heures	13	750 000	Indels.
Oxford Nanopore	Exonucléase par Nanopore	GridION MinION	4 - 10 millions 70 000	dizaines de milliers dizaines de milliers	96 70	variable 48 heures	quelques dizaines 0,132	variable 1 000	Indels.

TABLE 1: Récapitulatif des différentes technologies de séquençage. La précision est donnée en %, le débit en Gb, et le coût en $.

Outil	Structure de données	Erreurs corrigées	Nombre de $reads$ (longueur)	Espace mémoire	Temps	$reads$ corrigés
SHREC	Arbre des suffixes	subs.	1 090 946 (70)	1 500	183	88,56
HybridSHREC	Arbre des suffixes	subs. + indels	977 971 (178)	15 000	28	98,39
HiTEC	Table des suffixes	subs.	1 090 946 (70) 4 639 675 (70)	757 3 210	28 125	94,43
Fiona	Table des suffixes partielle	subs. + indels	977 971 (178) 2 464 690 (142)	2 000 3 000	15 32	66,76
Coral	Table de hachage	subs. + indels	977 971 (178)	8 000	5	92,88
RACER	Table de hachage	subs.	2 119 404 (75) 101 548 652 (457 595)	1 437 41 700	23 104	76,65 42,95
BLESS	Filtres de Bloom	subs. + indels	1 096 140 (101)	11	6	84,38
LoRDEC	Graphe de De Bruijn	subs. + indels	33 360 $reads$ longs (2 938) et 2 313 613 $reads$ courts (100)	960	10	85,78

TABLE 2: Récapitulatif des différentes méthodes de correction des *reads*. L'espace mémoire est donnée en Mo. Le temps est donné en minutes. Les *reads* corrigés sont donnés en %, et la valeur indiquée est une moyenne.

Outil	Structure de données	Erreurs prises en compte	Nombre de *reads* (longueur)	Espace mémoire (en Mo)	Temps (en min)	*reads* mappés (en %)
MAQ	Table de hachage	subs. + indels	1 000 000 (44)	1 200	331	92,53
MrsFAST	Table de hachage	subs.	1 000 000 (100)	20 000	169	90,70
MrsFAST-Ultra	Table de hachage	subs.	2 000 000 (100)	2 000	57	91,41

TABLE 3: Récapitulatif des différentes méthodes de *mapping* de *reads*. L'espace mémoire est donnée en Mo. Le temps est donné en minutes. Les *reads* mappés sont donnés en %, et la valeur indiquée est une moyenne.

Peu d'outils sont présentés ici, mais de nombreuses méthodes de $mapping$ de *reads*, n'utilisant pas de structure d'index sur les *reads*, existent et produisent de très bons résultats, aussi bien en espace et en temps, qu'en qualité de $mapping$.

Outil	Structure de données	Nombre de *reads* (longueur)	Espace mémoire (en Go)	Temps R1 (en ms)	Temps R2 (en ms)	Temps R3 (en ms)	Temps R4 (en ms)
GkA	Table des suffixes modifiée + Table des suffixes modifiée inverse + Table associant k-mer - nombre d'occurrences	42 400 000 (75)	20	16	25	25	0,1
CGkA	Table de suffixes échantillonnée + 3 vecteurs de bits	42 400 000 (75)	3 - 7	1203	28	1278	28
PgSA	Table des suffixes échantillonnée + Table auxiliaire d'information sur les *reads* et k-mers	42 400 000 (75)	1 - 4	70	58	70	58

TABLE 4: Récapitulatif des différentes méthodes permettant de traiter les 7 requêtes. L'espace mémoire est donné en Go. Le temps est donné en millisecondes. Les requêtes 5-7 sont exclues du comparatif, car non implémentées dans GkA et CGkA lors des tests réalisés dans le papier introduisant PgSA.

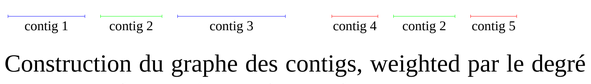
Correction de *reads* longs : Les *reads* NaS (Nanopore *Synthetic-long*)

Les nouvelles technologies de séquençage permettent de séquencer des *reads* de plus en plus longs, mais ceux-ci disposent d'un important taux d'erreur, avoisinant notamment les 30% pour les *reads* séquencés par la plateforme MinION. Comme le montrent les tableaux précédents, les méthodes de correction classiques ne sont pas adaptées à de tels *reads*, et sont donc très peu efficaces. Une solution alternative pour résoudre ce problème est la génération de *reads* dits synthétiques. Ces derniers sont générés via une approche hybride, utilisant des *reads* longs comme *templates* et des *reads* courts disposant d'un plus faible taux d'erreur. Les *reads* ainsi synthétisés, appelés *reads* NaS, car synthétisés à partir de *reads* de la technologie Nanopore, peuvent atteindre une longueur de 60 000, et s'aligner intégralement et sans erreurs. Nous présentons ici une première méthode permettant de synthétiser de tels *reads*, et la méthode que nous avons développé.

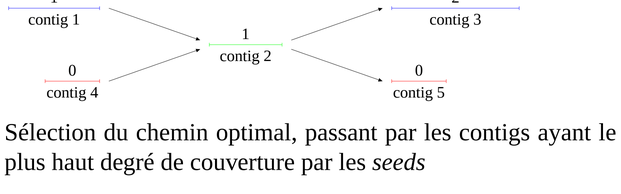
Première méthode

La première méthode de synthèse des *NaS* repose sur les étapes suivantes :

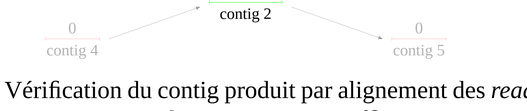
1. Alignement des *reads* courts sur le *read* long utilisé comme *template*, afin de trouver les *seeds*



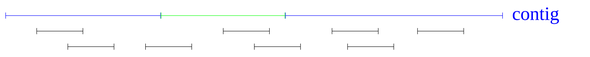
2. Recrutement de nouveaux *reads* courts, en recherchant des *reads* similaires aux *seeds*



3. Micro-assemblage de l'ensemble de *reads* obtenu



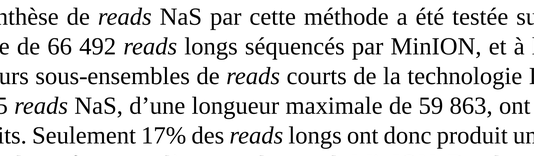
4. Obtention du *read NaS*



En général, un seul contig est produit par cette méthode, mais il est cependant possible que la phase de recrutement des *reads*, notamment dans les régions répétitives, recrute de mauvais *reads*, et que des contigs erronés, ne devant pas être associé au *template*, soient alors produits. Pour résoudre ce problème, et ne produire qu'un seul contig en sortie, il suffit d'employer la démarche suivante :

1. Obtention de plusieurs contigs

2. Construction du graphe des contigs, weighted par le degré de couverture des contigs par les *seeds*



3. Sélection du chemin optimal, passant par les contigs ayant le plus haut degré de couverture par les *seeds*

4. Vérification du contig produit par alignement des *reads* courts, et acceptation si la couverture est suffisante

Le temps de traitement d'un *read* long, et donc la synthèse d'un *read* NaS, prend en moyenne moins d'une minute, la majorité de ce temps étant provoquée par la méthode peu efficace de recrutement de *reads* similaires. Celle-ci est réalisée en indexant l'ensemble des k-mers de tous les *reads* non alignés dans un BFT, et en recherchant les k-mers des *seeds* dans celui-ci. Un *seed* et un *read* non aligné sont alors considérés comme similaires s'ils partagent au moins t k-mers ne se chevauchant pas.

La synthèse de *reads* NaS par cette méthode a été testée sur un ensemble de 66 492 *reads* longs séquencés par MinION, et à l'aide de plusieurs sous-ensembles de *reads* courts de la technologie Illumina. 11 275 *reads* NaS, d'une longueur maximale de 59 863, ont ainsi été produits. Seulement 17% des *reads* longs ont donc produit un NaS, ce qui est du au fort taux d'erreurs des *reads* MinION. De plus, 97% des *reads* ainsi synthétisés ont pu être alignés sur le génome de référence sans aucune erreur, prouvant ainsi l'efficacité de cette méthode.

Notre méthode

Notre méthode repose sur le même principe que la méthode précédente, mais vise à diminuer le temps d'exécution en proposant une méthode différente pour le recrutement de *reads*. Nous alignons donc tout d'abord les *reads* courts sur les *reads* longs *templates*, en se fixant un seuil $lmin$, et récupérons les *reads* :

- Totalement alignés
- Avec un préfixe de longueur $\geq lmin$ aligné
- Avec un suffixe de longueur $\geq lmin$ aligné

Pour chaque *reads* long *template*, les différents ensembles de *reads* sont ajoutés à trois listes, triées en fonction des positions de début d'alignements préalablement calculées. Notre méthode se décompose alors en deux phases. La première parcourt ces listes en parallèle, afin de recruter de nouveaux *reads* similaires aux *reads* totalement alignés, utilisés comme *seeds*, en considérant qu'un *read* est similaire à un *seed* si son préfixe (respectivement son suffixe) correctement aligné chevauche ce *seed* sur une longueur supérieure ou égale au seuil fixé $lmin$. Le processus de recrutement, met à jour la liste des *seeds*, afin de prendre en compte les allongements provoqués par les recrutements, et ainsi couvrir d'avantage le *read* long *template* considéré.

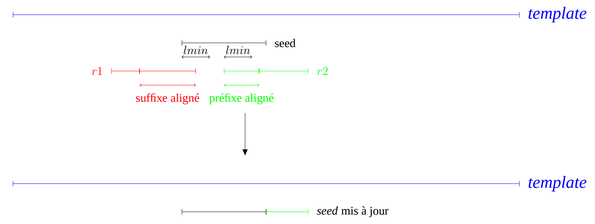


FIGURE 1: Illustration du processus de recrutement de *reads* similaires pour un *seed* donné. $lmin$ représente le seuil permettant de déterminer la similarité. Ici, on remarque que le *seed* est similaire à $r2$, mais pas à $r1$. $r2$ est donc recruté, et le *seed* est mis à jour avec les parties de ce *read* ne le chevauchant pas, afin d'étendre sa longueur.

La deuxième phase parcourt de nouveau les listes en parallèle, afin d'étendre les contigs produits par la phrase précédente. Nous nous affranchissons ici de la relation de similarité définie précédemment, définissions un seuil $lmax$, et recrutons alors un *read* partiellement aligné si son préfixe (respectivement son suffixe) chevauche le contig sur une longueur inférieure ou égale au seuil $lmax$.

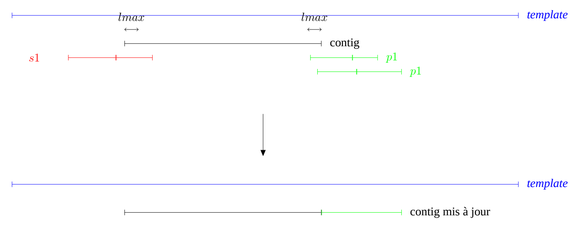


FIGURE 2: Ill.

Et la re du blabla pe ça passe