Rapport utilisateur

HULOT Valentin
GUTIERREZ-ANDRÉ Alban

Introduction

Dans ce rapport utilisateur, nous définirons tout d'abord ce qu'est un mapper et comment utiliser notre programme. Dans un second temps, nous parlerons des diverses optimisations que nous avons implémenté. Et, finalement, nous présenterons les résultats des tests que nous avons effectués pour évaluer le programme en fonction des paramètres en entrée.

Descriptif du programme

Le programme que nous avons implémenté lors de ce projet est un mapper. Le principe d'un mapper est d'aligner des séquences nucléiques sur un génome de taille largement supérieure en s'autorisant un certain nombre d'erreurs dans l'alignement. Ainsi le programme prendra en paramètre un génome de référence, une liste de séquences, les reads, un nombre d'erreur maximal à ne pas dépasser, appelé *dmax* et un fichier de sortie qui contiendra les résultats des alignements optimaux en fonction des reads. Il y a aussi un paramètre, appelé *k*, qui correspond à la taille des graines dans les reads. L'appel de fonction se fait comme suit sur ligne de commande : python minimapper.py -ref *fichier_reference*.fasta -reads *fichier_reads*.fasta -k *taille* -dmax *nombre_erreur_max* -out *fichier_sortie*.txt.

Le programme commence tout d'abord par faire une recherche de correspondance exacte entre les mots de taille *k* des reads et le génome. Ces mots de taille *k* sont appelés graines. Le paramètre *k* sert donc à définir la taille de ces graines. Ces graines permettent de limiter la recherche sur une portion du génome. Les résultats sont présentés comme suit :

nom_du_read *position_dans_genome* *brin* *nombre_erreurs*

La position dans le génome correspond à la position du meilleur alignement trouvé par le programme. La valeur de brin est soit +, ce qui signifie que l'alignement trouvé correspond à un alignement fait directement avec le read donné en entrée, soit –, ce qui signifie que c'est le reverse complément du read en entrée qui correspond à l'alignement. Le nombre d'erreurs correspond au nombre d'erreurs dans l'alignement optimal sachant que le read ne sera pas dans les résultats si le nombre d'erreurs dans l'alignement optimal est supérieur à *dmax*.

Les optimisations de temps de calcul

Une des optimisations que nous avons implémenté afin de réduire le temps de calcul est le stockage dans un tableau des positions dans le génome où nous avons déjà effectué un alignement en utilisant la programmation dynamique. En effet, une comparaison des deux séquences ayant déjà été faite, il n'est pas nécessaire de la faire une deuxième fois. Cette amélioration a permis un gain de temps considérable.

Une autre optimisation nous a également permis de diminuer drastiquement le temps de calcul. Nous avons décidés de ne remplir qu'une petite partie de la matrice au lieu de la remplir en entier, nous nous concentrons seulement sur une diagonale autour de la graine que nous avons trouvé en utilisant *dmax* comme borne et nous décidons de mettre les valeurs nécessaires au calcul autour de cette diagonale à *dmax*. Ainsi, comme l'algorithme ne renvois de résultat que si l'alignement a un score inférieur ou égal à *dmax*, nous ne perdons pas de précision tout en diminuant le temps de calcul.

Grâce à ces optimisations, nous avons réussi à faire chuter le temps de calcul sur un exemple concret de 200 secondes à 60 secondes (dans le jeu de test décrit plus bas, section *Tests sur l'influence des paramètres*, avec $dmax = 2$ et $k = 20$).

Les optimisations mémoire

La principale utilisation de la ressource mémoire vient du fait que l'on utilise de la programmation dynamique pour trouver le meilleur alignement pour une séquence donnée. Dans notre mapper, nous avons essayé d'implémenter cette étape afin qu'elle coûte le moins de mémoire que possible. La complexité mémoire d'un tel algorithme est de l'ordre de O(*tailleDuRead* x *tailleDuGénome*) et nous l'avons programmé de telle sorte que cela ne nous coûte que O(*tailleDuRead*), en ne stockant que deux colonnes de la matrice à la fois.

Tests sur l'influence des paramètres

Nous avons fait des tests pour des valeurs fixées de *dmax* en faisant varier *k*. En effet en fonction de la taille de ces ancres, les retours du programme peuvent s'éloigner d'un résultat optimal. Dans les tableaux suivants, la sensibilité correspond au pourcentage d'alignements trouvés par rapport à tous les alignements.

Résultats pour dmax = 0

k	Sensibilité (en %)	Temps (en secondes)
100	100	2
75	100	19
50	100	36
25	100	52

Les résultats ne sont pas surprenant car on prends ici des reads de taille 100 pour faire le test. La sensibilité sera toujours à 100 % car on veut un alignement parfait entre le read et la partie du génome. Ainsi il vaut mieux prendre comme valeur pour *k* 100, afin de minimiser le temps de calcul.

De manière générale, si on utilise ce programme avec *dmax* valant 0, alors il vaut mieux directement poser *k* tel qu'il soit égal à la taille des reads en entrée.

Résultats pour dmax = 2

k	Sensibilité (en %)	Temps (en secondes)
100	14	2
75	45	20
60	71	31
50	88	39
35	99	51
34	100	53
20	100	62

On remarque ici principalement deux choses que l'on va retrouver dans les autres tests, plus *k* augmente et plus la sensibilité ainsi que le temps diminuent. Nous avons mis en évidence la valeur de *k* où la sensibilité passe à 100 % (ici il vaut 34).

Résultats pour dmax = 4

k	Sensibilité (en %)	Temps (en secondes)
100	7	2
75	27	21
50	71	42
30	99	66
26	99	71
25	100	70
20	100	74

Nous avons mis en évidence la valeur de *k* où la sensibilité passe à 100 % (ici il vaut 25).

Résultats pour dmax = 8

k	Sensibilité (en %)	Temps (en secondes)
100	5	3
75	19	24
50	56	49
40	78	62
30	95	76
21	99	88
20	100	91

Nous avons mis en évidence la valeur de *k* où la sensibilité passe à 100 % (ici il vaut 20).

Résumé des résultats

Comme dit précédemment, on remarque au fil des tests que plus *k* augmente plus le temps de calcul et la sensibilité diminuent. De plus, on remarque une borne sur *k* à partir de laquelle on obtient toujours un résultat de sensibilité de 100 %. Cette borne varie en fonction de la valeur de *dmax*, plus *dmax* est grand et plus cette borne sera petite. Ceci est dû au fait que plus on autorise d'erreur, plus la recherche de graines de taille trop importante dans le génome de référence risque d'être inadéquate.

Ainsi, tout dépend du but recherché : si on veut un résultat optimal il faut prendre un *k* assez petit mais si on désire juste avoir une partie de la solution, on peut le choisir plus grand. Dans les tests effectués, on remarque que la valeur 20 pour *k* donne toujours un résultat optimal, mais pour des valeurs de *dmax* plus grandes ce ne sera plus le cas.

Conclusion

Le fait d'avoir en retour un résultat optimal, n'est pas garanti car il dépend en grande partie des paramètres choisis en entrée, mais, en contrepartie, le temps de calcul est allégé.