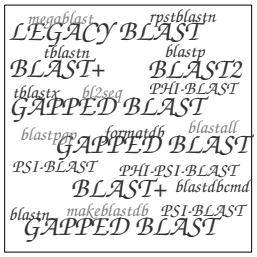


La galaxie BLAST

20 ans d'utilisation de la méthode BLAST

Guillaume Fernandez
Ancien étudiant en génétique de l'université Paris-Sud 11
Ancien étudiant en biofinformatique génomique à l'université Paris 7

Manuel rédigé en Octobre 2010, mis à jour en Mai 2013

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir®

université PARIS DIDEROT PARIS 7

Document initialement rédigé à l'université Paris-7 en Master 2 de Bioinformatique – Octobre 2010

Auteur : Guillaume Fernandez
Numéro d'étudiant : 21005696

Unité de génomique
Responsable : Dr. Gaëlle Lelandais

Le document initial de 2010 a obtenu la note de 18/20.

université PARIS DIDEROT PARIS 7

Je remercie David Lipman et Thomas Madden pour leurs réponses à mes questions[1].

The BLAST programs are unlikely to remain static, and there are many possible avenues for future improvement[2]

Altschul, Madden, Schäfer, Zhang, Zhang, Miller et Lipman, 1997 [3]

1 I would like to thank Dr. David Lipman and Dr. Thomas Madden for the time they spend answering my questions.
2 Les programmes BLAST ne resteront pas statiques et il existe de multiples voies d'amélioration pour le futur.

Adresse de contact : guillaume.fernandez (*at*) alumni.u-psud.fr

Sommaire

Mots : 14523 Caractères : 93476 Pages : 47

Addenda

Clamart, le 21 mai 2013. Ces pages ont été insérées après la date de compilation initiale du mémoire (en 2010).

A) Dernières annonces faites par le NIH/NCBI

Consulter l'Annexe 8 : dernières annonces du NCBI, BLAST+ 2.2.28, arrêt de Blastcl3, la collection « nt » par défaut comme base de recherche nucléotidique, page 42. Cette annexe a été ajoutée en mai 2013.

B) Erratum

1) Page 11 (page 9 de la version publiée en 2010)

Après « Pour l'ADN, la liste des mots qui sera comparée aux séquences de la banque interrogée est plus simple [1]. C'est la liste de tous les mots contigus de longueur w de la requête, en prenant w=12 par défaut. » a**jouter :** « Le blast nucléotidique recherche toute correspondance exacte (exact match) entre les mots de la requête et de la banque interrogée [23] à la différence du blast protéique vu plus haut qui utilise une liste de mots voisins en utilisant le seuil de score de voisinage T (l'option threshold) [22] avant de passer à la phase d'extension suivante qui produit le HSP. »

2) Page 12 (page 11 de la version publiée en 2010)

« Par défaut T est passé de 13 à 11 dans BLAST 2 pour blastn [15],[3] » doit être **remplacé par** « par défaut T est passé de 13 à 11 dans BLAST 2 pour blastp [15],[3] ». En effet, l'option *threshold* n'est disponible que pour blastp, blastx, tblastn et tblastx [17].

Après « BLAST 2 n'invoque une élongation sans brèche de l'alignement si et uniquement si :

- deux touches sont trouvées sur la même diagonale ;
- les deux touches ne se recouvrent pas ;
- les deux touches sont séparées au plus par une distance A. »

ajouter « BLAST 2 n'invoque une élongation sans brèche de l'alignement si deux touches sont trouvées sur la même diagonale pour blastp et dc-megablast uniquement. L'option de

réglage de la distance A, appelée window_size n'apparait en effet que pour ces deux programmes [17]. Blastn conserve la touche (hit) unique avant de créer par extension le HSP [22], puis l'élongation avec trous (gap) par programmation dynamique. ».

Ajouter également « La distance A est de 40 par défaut [22] pour l'ensemble des programmes où l'option window_size est disponible. »

C) Pièces jointes ajoutées au dossier (références et annexes)

Discontiguous MEGABLAST is better at finding nucleotide sequences similar, but not identical, to your nucleotide query.
http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=ProgSelectionGuide#blastn
Consulté le 21.5.2013. Il s'agit d'un extrait de *Blast Program Selection Guide*, déjà cité en référence [22].

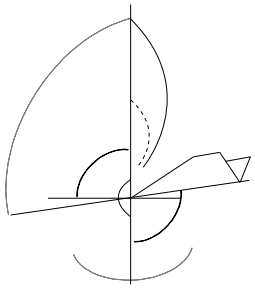
How does blastall find the alignment?
http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/blastall/blastall_node4.html
Consulté le 21.5.2013. Il s'agit d'un extrait de la documentation du Dr. Tao déjà cité en référence [20].

23 : What is discontiguous Mega BLAST ?
http://www.ncbi.nlm.nih.gov/blast/discontiguous.shtml
Consulté le 21.5.2013. Nouvelle référence.

24 : Alignement de séquences , version 8
Il peut être utile de consulter ce précis sur les méthodes d'alignement rédigé séparément pour le chapitre « Systèmes de score » sur les matrices PAM et BLOSUM.
http://gf3.myriapyle.net/gcde/data/alignement-sequences-8-ferndz.pdf
G. Fernandez, Mars 2009, faisant référence à [25, 26]

25 : Henikoff S., Henikoff J.G., « Amino acid substitution matrices from protein blocks », PNAS , November 15, vol. 89 no. 22 10915-10919, 1992.

26 : M.O. Dayhoff, ed., « Atlas of Protein Sequence and Structure », Vol5, 1978.



Addenda *

Introduction

BLAST (acronyme de *Basic Local Alignment Search Tool*) est un ensemble de programmes qui permettent de trouver dans une banque les séquences proches d'une séquence soumise grâce à une approche heuristique [1]. C'est un outil de recherche d'alignements locaux basique... ...et très populaire. Il a été originellement publié en 1990 par les chercheurs du NCBI (*National Center for Biotechnology Information*). Une interface web est disponible à l'adresse suivante : `http://www.ncbi.nlm.nih.gov/BLAST/`. Pour une utilisation en ligne de commandes, BLAST est également téléchargeable depuis les serveurs du NCBI[3]. Cette introduction montre dans quelle perspective scientifique ces programmes peuvent être utilisés et annonce les premiers mot-clés à connaître pour la compréhension de leurs modes d'action.

a) Pour une prise de vue : le contexte scientifique

Les technologies de séquençage ont permis de produire des lectures partielles de génomes dès 1977 (année de publication de la technique Sanger du séquençage). Un nouveau problème s'est alors posé à la communauté scientifique. Comment stocker, organiser et distribuer l'information ? La réponse a été apportée avec les banques et les bases de données[4]. C'est par exemple en 1980 qu'ont été créés l'EMBL et GenBank. Les séquences contenues dans les banques contiennent entre autres des cadres de lectures et des régions régulatrices. L'annotation d'un génome est le processus qui vise à ajouter à la séquence brute une information supplémentaire : la localisation des cadres codants et des régions régulatrices, les fonctions associées à chaque ARN ou protéine produite [9]. La fiche GenBank d'un génome complet de phage est présentée en annexe 1 à la page 34. Ce génome est annoté. En effet, le lecteur de la fiche est capable de lire l'emplacement de gènes qui ont été repérés[5] dans le génome par des annotateurs professionnels ou des procédés d'annotation automatique. Des logiciels tels que Artemis[6] permettent de créer ces fiches.

3 Adresse de téléchargement : http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download. Le logiciel est dans le domaine publique, les sources peuvent être téléchargées. Cependant, il existe certains dérivés commerciaux (AB-BLAST).

4 Le journal Nucleic Acids Research effectue tous les ans une revue des banques et bases disponibles. On peut accéder à la liste ainsi produite à l'adresse http://www.oxfordjournals.org/nar/database/c/.

5 La signature d'un gène est constituée par un signal de début, un signal de fin, une composition statistique particulière en bases nucléotidiques, la présence de sites marqueurs d'exons et d'introns pour les eucaryotes.

6 Artemis est disponible au téléchargement à l'adresse http://www.sanger.ac.uk/resources/software/artemis/.

b) Recherche de similarité dans les banques de données

Une bonne façon de réaliser une annotation d'une séquence dont on ne connaît que la succession des bases nucléotidiques est de chercher si cette séquence est déjà stockée dans une banque, ou dans le cas contraire, de chercher si une séquence similaire a déjà été décrite dans un organisme. Cette approche est dite « par similarité ». Une similarité de séquence peut être le fait d'une homologie résultant de l'évolution ou d'une analogie fonctionnelle. On part sur le principe qu'un gène similaire à une séquence requête peut avoir la même fonction, ou une fonction proche. Cette fonction a pu être héritée chez différents organismes depuis un ancêtre commun, ou bien être le fait d'une convergence. Dans une perspective phylogénétique, l'expérimentateur devra faire manuellement la distinction, parmi les séquences récoltées par similarité, entre celles qui sont réellement orthologues et celles qui résultent d'une convergence évolutive. Quoiqu'il en soit l'approche par similarité permet d'avoir une idée sur la fonction d'une protéine codée par un gène si la fonction d'une séquence

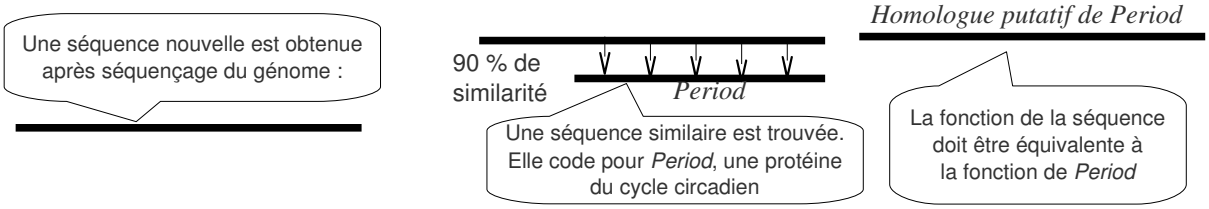


Figure 1 : l'approche par similarité peut être représentée en trois étapes, de la gauche vers la droite. 1 - une séquence de fonction inconnue est échantillonnée (séquençage du génome, séquençage d'un environnement). 2 - On trouve une séquence similaire dans une banque de données. La fonction de cette séquence aura pu être déterminée expérimentalement. 3 - On suppose que la séquence inconnue a une fonction similaire à la séquence dont on connait la fonction. Seule l'expérimentation *in vivo* confirmera la fonction de la protéine.

Évidemment, l'approche par similarité n'est pas sans danger. Des erreurs d'annotation contaminent souvent l'ensemble d'une base de données (*cf.* figure 2) [9]. Seule la vérification expérimentale est à même de corriger une annotation.

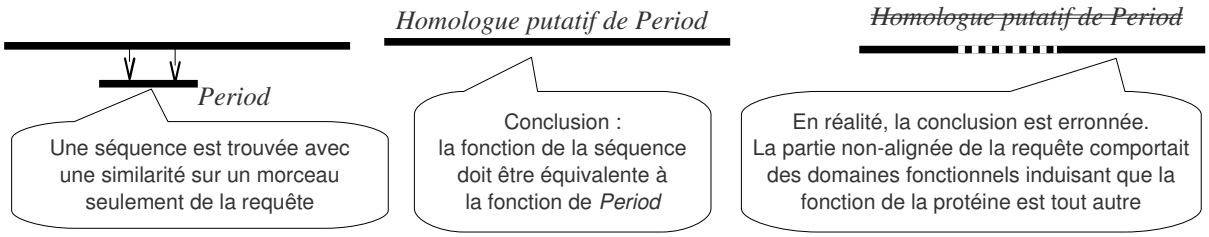


Figure 2 : Origine d'une contamination d'une banque de données par une annotation erronée due à l'approche par similarité. Si la mauvaise annotation n'est pas corrigée, toutes les séquences qui seront ensuite étudiée par similarité à cette séquence mal annotée propageront l'erreur.

c) *La galaxie BLAST*

BLAST est un ensemble de programmes de recherche de séquences par similarité. Le programme a fait l'objet de constantes améliorations depuis 1990, aboutissant à un corpus d'articles étoffé. Le réseau des programmes (des plus anciens aux versions récentes), des articles associés, de la documentation, des serveurs qui mettent à disposition leur propre version de BLAST tout autour du monde[7] et enfin des utilisateurs constitue une véritable galaxie. Le document princeps (*Basic Local Alignment Tool*, 1990 [1]) a été cité plus de 15 000 fois entre 1982 et 2002 [11], ce qui le place en troisième position des articles les plus lus sur cette période.

BLAST permet de rechercher par similarité des séquences nucléiques sur une banque de données nucléique, des séquences protéiques sur des banques de protéines et différentes combinaisons de ses deux premières possibilités en utilisant la traduction du code génétique en acides aminés (appliquée soit à la séquence requête, soit à la banque interrogée, soit aux deux) [1].

Les principaux mots-clés à connaître pour son emploi sont :

- la requête (*query*) : la séquence soumise au programme dont on souhaite trouver des similarités ;
- le sujet (*subject*) : toute séquence tirée de la banque de données cible ;
- la touche (*hit*) : similarité significative identifiée entre la séquence requête et une séquence cible ;
- les paires de score élevé (*HSP* ou *High Scoring Pairs*) : paire de séquence présentant une forte similarité ;
- l'E-Value (ou *Expected-Value*) : nombre d'occurrences d'une touche du fait du pur hasard.

L'objectif de cette synthèse est de présenter les évolutions de BLAST, publiées depuis 1990 par les chercheurs du NCBI et de caractériser l'utilisation qui peut être faite de ces programmes dans le cadre de la recherche en biologie.

> Note sur la démarche de recherche de l'information pour la présente synthèse
>
> Les deux articles graines de BLAST de 1990 [1] et 1997 [3] ont constitué le point de départ de l'analyse. Par la suite, l'application pratique de l'algorithme décrit dans ces articles a été recherchée, notamment en réalisant des tests utilisant les différents arguments des programmes BLAST. Pour cela la documentation en ligne sur

7 Sur ParameciumDB (http://paramecium.cgm.cnrs-gif.fr/tool/blast), sur Fly-Base (http://flybase.org/blast/), sur l'EBI (http://www.ebi.ac.uk/Tools/blast/), sur Ensembl (http://www.ensembl.org/Multi/blastview)

le serveur du NCBI a été sollicitée [10] et le contact a été pris avec Thomas Madden pour éclairer des points de détails. Enfin, de nombreux autres articles émanant du NCBI ont permis d'acquérir une vision évolutive des programmes, ainsi que de leur utilisation. A chaque nouvelle publication les auteurs utilisent un exemple biologique, ce qui permet d'employer ces articles à la fois pour étudier l'algorithme et l'utilisation qu'il est possible d'en faire. Le corpus d'articles accompagnant les programmes BLAST est dissociable en quatre parties : les articles traitant des méthodes statistiques d'évaluation d'alignements de séquences similaires [2], [7] ; les articles écrits par les chercheurs du NLM à Bethesda décrivant les programmes utilisant ces méthodes [1], [3], [6] ; les articles dits de « vulgarisation » qui visent à expliquer BLAST en termes simples au biologiste ([5], [8], [12], [16]). La documentation en ligne constitue enfin une quatrième et essentielle part de la littérature ([10], [14], [15], [18], [22]).

La méthode BLAST : évolution et utilisations

I - Les origines de BLAST : tout commence en 1990

Altschul, Gish, Miller, Myers, et Lipman publient l'article fondateur en 1990. Ils présentent BLAST comme une nouvelle approche pour des comparaisons rapides de séquences. Selon eux, l'algorithme est simple et robuste. Il est beaucoup plus rapide que les programmes existants (Smith et Waterman, 1981) pour une sensibilité comparable et peut être couplé à des résultats mathématiques récents (en 1990) permettant d'évaluer la significativité des alignements produits [1]. En effet, BLAST trouve des sujets similaires à une requête, aligne chaque sujet avec la requête et donne un score d'alignement. Le score est une mesure de la qualité de l'alignement [1].

Par exemple, pour les deux séquences suivantes de longueurs L1=13 et L2=7,

```
S1: GGATACCTGACCA
S2: TAGATCA
```

si l'on obtient l'alignement *local* suivant de longueur 7 :

```
N° de colonne :              1234567
Séquence 1          :    GGATACCTGACCA
Séquence 2          :       TAGATCA
```

le score de l'alignement est pénalisé les évènements de différence non synonymes (trois événements) et est favorisé par les évènements d'identité ou de différence synonyme[8] [1]. Le score serait maximal en cas d'identité parfaite de la requête et du sujet.

Les pénalités et récompenses attribuées aux évènements calculés par un alignement sont réglées en utilisant une matrice de substitution. L'objectif est de construire des systèmes de score tenant compte de la réalité biologique. Pour les protéines, il existe des matrices fondées sur les caractéristiques physico-chimiques des acides aminés et des matrices exploitant l'évolution moléculaire des protéines. Les secondes sont construites à partir de substitutions observées entre acides aminés au cours de l'évolution moléculaire, déterminées à partir d'alignements multiples de protéines homologues. Les fréquences ainsi mises en évidence sont représentatives des probabilités de substitutions altérant peu les fonctions de la protéine. La connaissance de ces probabilités permet de construire des matrices de substitutions

8 Une différence synonyme n'est possible que pour les protéines, où à la troisième base d'un codon.

adaptées à la mise en évidence de relations d'homologie. Deux grandes catégories de matrices se sont dégagées : les PAM et les BLOSUM .

Les auteurs évaluent l'utilité de leur méthode en utilisant les résultats théoriques sur la distribution des scores de paires à score élevé issues de la comparaison de séquences aléatoires [1], résultats publiés seulement quelques mois auparavant [2]. L'idée est de donner pour chaque sujet non-seulement le score mais une mesure statistique du nombre de touches auxquels on peut s'attendre du fait du simple hasard. Cette mesure statistique peut s'écrire sous cette forme :

$E = m.N.K.e^{-\lambda .S}$ [1], [13], où m est la longueur de la séquence requête, N la taille de la banque (en nombre de nucléotides ou d'acides aminés), K et λ deux constantes déterminées en fonction des critères de similarités des alignements [2] et S le score de l'alignement entre requête et sujet.

Interprétation. L'obtention d'un alignement de score élevé a très peu de chances d'être le fait du simple hasard. Lorsque le score augmente, l'E-value diminue exponentiellement. Lorsque le sujet et la requête sont identique l'E-value vaut 0 : il y a 0 chance pour que l'alignement soit le fait du hasard et non d'une similarité biologique.

1) Algorithme de BLAST (version de 1990)

BLAST va tenter d'établir des touches (*hits*) entre la requête et des séquences de la banque de données (figure 3). Le nombre de caractères (nucléotides ou acides aminés) de ces touches est déterminé par le paramètre *w*.

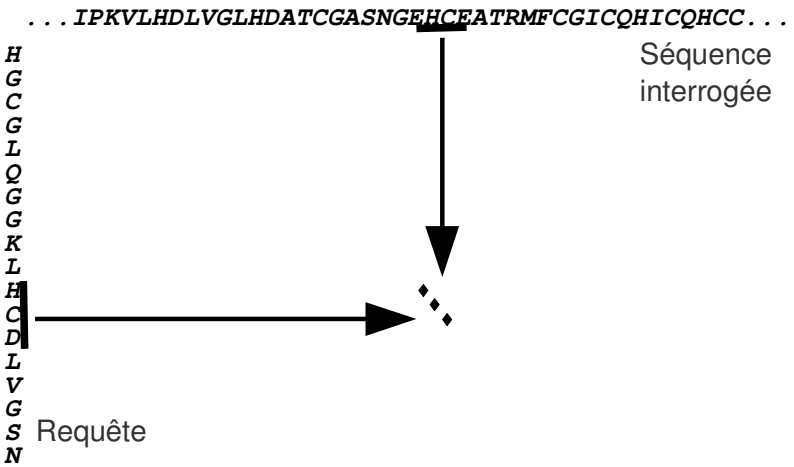


Figure 3 : l'objectif est de trouver des correspondances de taille *w* entre la requête et les séquences de la banque de données.

Pour les protéines, le programme établit *a priori* une liste de tous les mots possibles de longueur *w*. Par défaut, *w*=3 [15]. Cette opération est faite sans regarder aucune séquence [1]. Ensuite, pour chacun des mots de trois

lettres de la séquence, le programme met à jour la liste pour ne conserver que les mots voisins des mots effectivement trouvés dans la séquence (figure 4). Les mots qui ont un score inférieur à un seuil de similarité *T* avec les mots de la séquence ne sont pas considérés comme voisins [1].

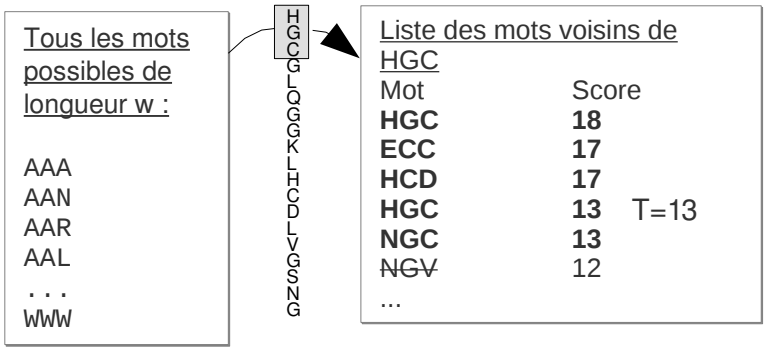


Figure 4 : mots voisins du premier mot de la requête. Parmi tous les mots de longueur *w* qu'il est possible de former, seuls certains sont similaires au dessus d'un score seuil *T* (seuil de similitude du mot). Le programme dresse la liste de ces mots voisins pour chacun des mots de la requête. La requête est une séquence protéique, *w*=3 et *T*=13. Les scores des mots sont inventés pour l'exemple. La seconde liste qui sera dressée sera la liste des mots voisins de GCG et ainsi de suite jusqu'au dernier mot.

Ensuite, chacun des mots voisins est confronté à tous les mots de longueur w de toutes les séquences de la banque. Chaque fois qu'une identité est trouvée, une touche est enregistrée (voir la figure 5). Finalement, on voit que grâce à l'utilisation du seuil de voisinage *T*, BLAST ne conserve que les paires de mots qui s'alignent au moins avec un score de *T* [1].

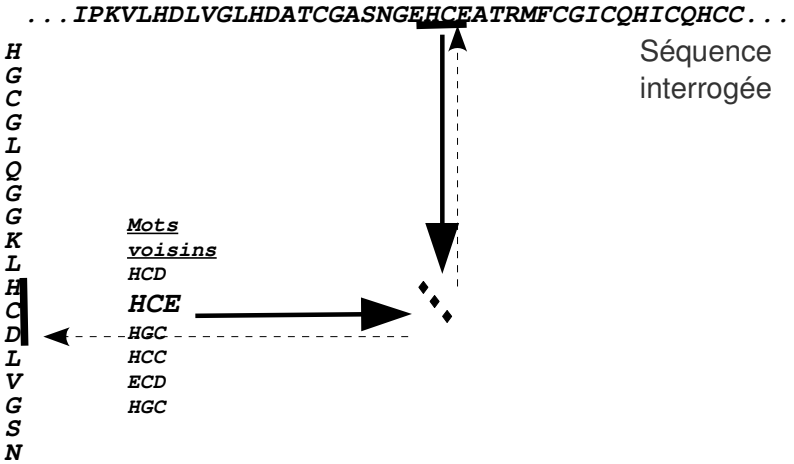


Figure 5 : Un des mots voisins (HCE) est trouvé identique dans une séquence de la banque interrogée (flèches grasses). Alors, un *hit* est enregistré pour cette position (flèches discontinues). Les mots voisins sont hypothétiques car inventés pour l'exemple.

Ensuite, le programme réalise une extension sans brèche de l'alignement dans les deux sens autour des mots courts identiques (fig. 6). Le score de l'extension est calculé pas à pas lors de chaque ajout d'une nouvelle paire de caractères à la suite de l'alignement. L'extension s'arrête

quand elle fait baisser d'une quantité donnée le score maximal déjà enregistré durant celle-ci [1]. Cette quantité est appelée X_{stop} (*X drop-off* dans la littérature [3]).

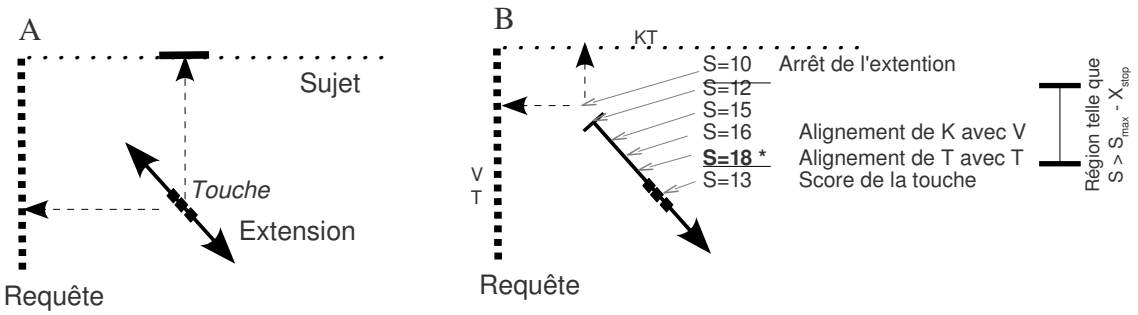


Figure 6 : A - Extension de l'alignement autour de la touche initiale. B – chaque paire d'acide aminé a son propre score (par exemple, aligner T avec T rapporte *+5* selon la BLOSUM 62 mais aligner K avec V rapporte *-2*), modifiant ainsi le score global *S* de l'alignement. Le meilleur score enregistré pendant l'extension est S=18. A un moment, le score passe de X_{stop} en dessous de 18. Si X_{stop} = 9, l'extension est arrêtée à juste avant que S n'atteigne la valeur 11 (18-9 = 11).

Pour l'ADN, la liste des mots qui sera comparée aux séquences de la banque interrogée est plus simple [1]. C'est la liste de tous les mots contigus de longueur *w* de la requête, en prenant *w*=12 par défaut.

2) Utilisation de BLAST

BLAST se présente sous la forme d'un programme avec plusieurs options. L'option `blastn` est utilisée pour chercher des séquences nucléiques avec une requête nucléique. `blastp` est utilisé pour les protéines. `blastx` sert à comparer une requête en acide aminée traduite en protéine à une base de données de protéines. `tblastn` sert à comparer une requête de protéine sur un banque nucléique traduite. `tblastx` sert à comparer une requête nucléique traduite à une banque nucléique traduite. Pour plus d'informations, le centre d'apprentissage du NCBI est très instructif [20].

Un résultat typique de BLAST dans sa version de 1990 est de retourner plusieurs segments alignés pour un seul couple requête-sujet, suggérant, sans le montrer explicitement, que l'on pourrait obtenir un alignement plus large en incorporant des trous dans celui-ci, modélisant des insertions ou des délétions (figure 7).

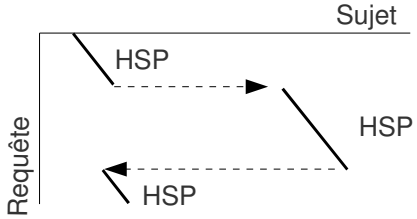


Figure 7 : plusieurs alignements (HSP) sont produits par sujet rapatrié (traits pleins) – BLAST de 1990 uniquement. Cela suggère que l'on pourrait les relier (traits discontinus) pour modéliser des insertions ou des délétions dans la requête et le sujet.

II - Les brèches : la révolution de 1997

En Juillet 1997 est publié un article présentant une mise à jour majeure de l'algorithme de BLAST (*Gapped BLAST and PSI-BLAST : a new generation of protein*[9] *database search programs*, en référence [3]). Le 19 Août 1997 la version 2.0 de BLAST est rendue disponible sur le serveur du NCBI ([4], annexe 3 page 36). La recherche des paires de mots qui vont être étendues est modifiée, des brèches peuvent être introduites dans les séquences et la possibilité d'élargir la recherche de séquences en réinjectant le résultat d'un BLAST dans une nouvelle étape de recherche est ajoutée (BLAST itératif). Par souci de commodité ce nouvel ensemble sera dénommé « BLAST 2 » ci-après.

A. La méthode des deux touches[10] restreint le nombre d'extensions

La phase d'extension occupe jusqu'en 1997 plus de 90% du temps d'exécution de BLAST [3]. L'idée de cette première modification de l'algorithme est de réduire le nombre d'extensions déclenchées pour les réserver aux cas où la probabilité est plus élevée d'obtenir un bon résultat. Un segment contenant des paires alignées avec un score élevé (*High Scoring Pairs*) paraît être en effet plus long qu'une simple paire de mots et peut donc contenir plusieurs de ces paires de mots sur la même diagonale et à des distances pas trop éloignées les unes des autres [3].

BLAST 2 n'invoque une élongation sans brèche de l'alignement si et uniquement si :

- deux touches sont trouvées sur la même diagonale ;

- les deux touches ne se recouvrent pas ;

- les deux touches sont séparées au plus par une distance A.

Pour cela, BLAST enregistre dans un tableau les coordonnées de chaque touche et calcule les distances à partir de ce tableau. Ces conditions sont représentées dans la figure 8A en page 13.

Pour conserver une sensibilité comparable, la valeur par défaut de T (seuil de similarité des mots voisins) doit être abaissée par rapport au BLAST originel[11] [3]. Ainsi, plus de mots initiaux seront alignés, mais moins d'entre eux seront étendus.

9 Les programmes BLAST permettant l'alignement de nucléotides font aussi l'objet d'une mise à jour.
10 « The two-hit method » *in* [3].
11 Par défaut T est passé de 13 à 11 dans BLAST 2 pour blastn [15], [3].

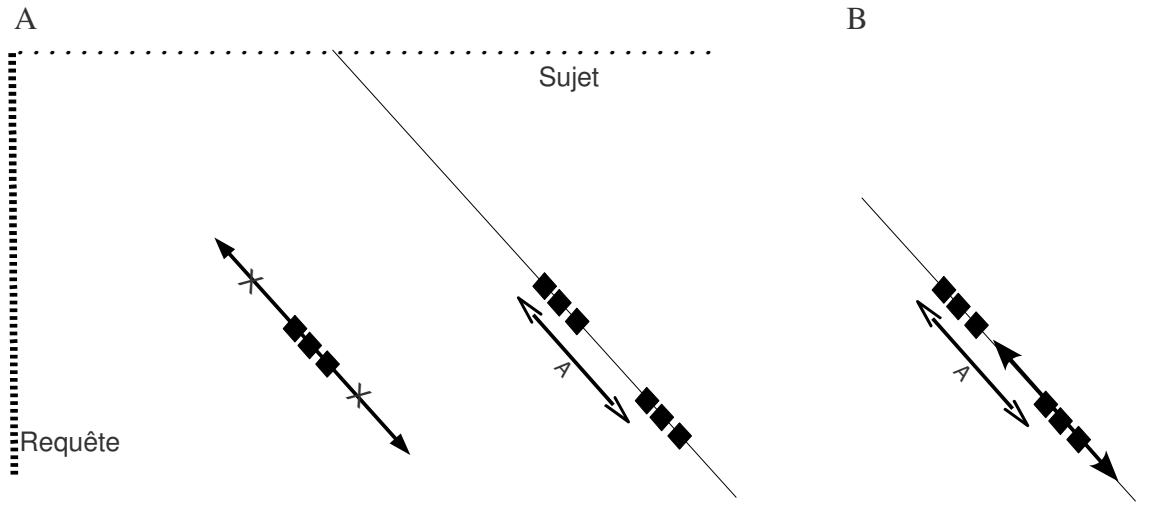


Figure 8 : A - BLAST 2 n'étend plus systématiquement chaque touche (*hit*). Une extension n'est déclenchée que pour deux touches ne se recouvrant pas séparées au plus de A sur la même diagonale. B – Déclenchement de l'extension sans brèche à partir du second *hit* [3].

L'extension sans brèche démarre à partir de la seconde touche (fig. 8B). L'extension s'arrête quand le score décroit en dessous de X du meilleur score observé jusque-là. Le résultat est un premier segment à score élevé de paires alignées (HSP). Sous certaines conditions, ce segment va ensuite être étendu par une extension avec brèches.

B. BLAST 2 inclus des brèches dans l'alignement des molécules

Prenons les deux séquences suivantes de longueurs L1=12 et L2=6,

```
S1: GGATACTGACCA
S2: TAGATA .
```

En autorisant des brèches (*gap*) on peut obtenir l'alignement *local* suivant de longueur 7 :

```
N° de colonne :          1234567
Séquence 1       :    GGATAC-TGACCA
Séquence 2       :       TAGAT-A
```

Cet alignement modélise les évènements qui ont pu survenir au cours de l'histoire évolutive de la requête et du sujet. Ainsi, le trou inséré en position 4 de l'alignement dans la séquence 1 peut représenter un événement de délétion d'une nucléotide dans cette séquence, ou bien d'insertion dans la séquence 2. Il est biologiquement bien adapté de recourir à une solution de recherche par similarité permettant l'incorporation des trous dans les séquences recherchées (ou dans la requête).

Ici, le score de l'alignement est pénalisé les évènements de trous (2 évènements) et de différence non synonymes (un évènement) et est favorisé par les évènements d'identité ou de différence synonyme.

Pour qu'une extension avec brèches soit déclenchée, le HSP généré à l'étape précédente doit posséder un score supérieur au seuil S_g. S_g est choisi automatiquement tel que pas plus d'une extension est invoquée pour 50 séquences de la base de donnée [3]. Une extension avec trous demande 500 fois plus de temps qu'une extension sans trou. Cependant, en réalisant seulement un nombre très limité d'extensions avec brèches, la fraction de temps demandé pour leur exécution reste faible [3].

Partant d'un segment étendu avec un score supérieur à S_g, le programme doit choisir une graine, c'est-à-dire une paire de résidus initiale, pour commencer la nouvelle extension. Cette extension est réalisée en utilisant la programmation dynamique[12]. Le programme localise le score de longueur 11 qui possède le score le plus élevé dans le HSP. Il utilise ensuite le résidu central comme graine de l'extension (voir la figure 9) [3].

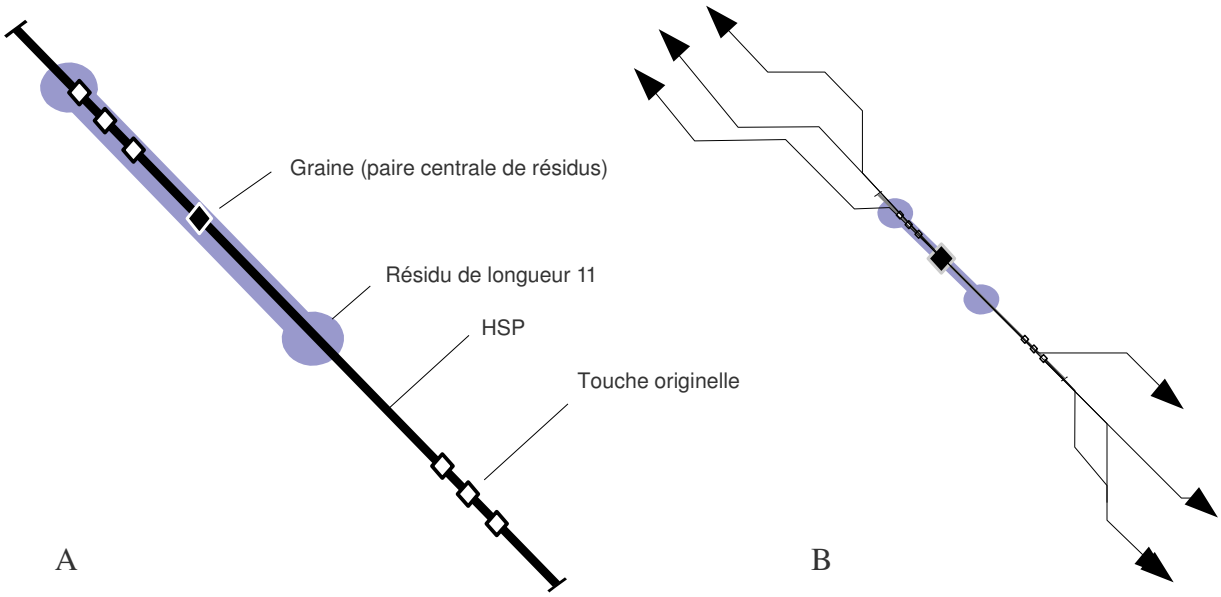


Figure 9 : A – définition de la graine initiale utilisée pour l'extension avec brèches. Explications dans le texte. B – La programmation dynamique évalue les différents chemins possibles et ne retiendra que le meilleur lors de la « remontée » des opérations effectuées.

En réalité, l'élongation avec trous est réalisée en deux étapes [3]. Il existe un alignement avec trous « initial » qui calcule uniquement le score et l'étendue de l'alignement. Cette information est stockée, et les différents alignements (dits préliminaires [17]) avec trous obtenus avec une requête sont rangés par ordre de score. Il est alors possible d'éliminer ceux qui recouvrent un alignement de plus grand score

12 Ainsi la nouvelle version de BLAST cantonne-t-elle la partie heuristique de son algorithme à la recherche des touches initiales. L'heuristique de BLAST 2 sert à confiner l'alignement par programmation dynamique entre une requête et un sujet à certaines régions. BLAST 2 demeure bien une méthode approchée : il produit un résultat localement optimal, rien de moins, rien de plus.

déjà trouvé. A cette étape, nous n'avons pas encore enregistré l'alignement complet (Thomas Madden, communication personnelle – voir l'annexe 2).

L'alignement final avec trous (présenté à l'utilisateur) utilise un Xg plus grand [3] pour augmenter la précision de BLAST dans la production d'un alignement local optimal et enregistre réellement le tracé comprenant les indels (Thomas Madden, communication personnelle) à partir des alignements trouvés à la phase précédente [17].

En résumé, le temps d'exécution est optimisé en deux endroits : lors de la décision d'extension sans brèche (méthode de la double touche) et lors de la décision d'extension avec brèches (méthode du score minimal S_g).

Utilisation en ligne de commande

Pour installer BLAST 2 sous Ubuntu[13], utiliser la commande `apt-get install blast2` et télécharger le fichier au format fasta qui constituera la banque locale sur laquelle vous allez travailler. Pour formater cette banque, on applique la commande `formatdb -i monfichier.fasta -p F -t 'ma banque' -b F -n mabanque`. Pour lancer un blastn sur cette banque à partir d'une séquence `seq.fasta` il faut ensuite appliquer la commande `blastall -p blastn -i seq.fasta -d mabanque -o resultat.txt`.

Options de formatdb

-i : indique le fichier d'entrée (entrer plusieurs fichiers est possible)
-p : indique s'il s'agit ou non de séquences protéiques (T pour True, par défaut F pour False)
-t : donne un titre écrit dans les entêtes de la banque
-b : indique si le fichier d'entrée est au format ASN.1 (T, par défaut F)
-n : nomme le fichier de sortie (obligatoire si c'est une concaténation de plusieurs fichiers d'entrée).

Options de blastall [15]

- p : choix du programme, blastn, blastp, blastx, tblastn ou tblastx
- i : pour donner le fichier d'entrée. Le fichier d'entrée peut contenir une seule séquence ou plusieurs séquences au format fasta.
- L, pour chercher avec la portion de 100 à 400 d'une requête, utiliser : L "100,400"

13 La version 2.0 a depuis été améliorée. De nouveaux programmes remplacent maintenant les anciens à partir de la version 2.2.21. Sous Ubuntu d'avril 2009 (système de l'auteur) le paquet distribué est le 2.2.18 et contient la version originelle (*Legacy BLAST*). Sinon, *Legacy BLAST* est toujours disponible au téléchargement sur le serveur du NCBI, même celui-ci recommande l'utilisation des nouveaux logiciels (vus plus avant dans cette synthèse).

- d : pour donner le nom de la banque interrogée (locale ou à distance).
- G : coût d'ouverture de gap (par défaut 5 pour blastn, 10 pour blastp, blastx et tblastn)
- E : coût d'extension de gap (par défaut 2 pour blastn, 1 pour blastp, blastx et tblastn)
- M : matrice de correspondance à utiliser pour les protéines . (au choix : BLOSUM62, BLOSUM45, BLOSUM80, PAM30, PAM70).
- q : pénalité pour un mismatch de nucléotide lors d'un blastn (3 par défaut)
- r : récompense pour une concordance nucléotidique exacte lors d'un blastn (1 par défaut)
- W : taille du mot (0 invoque le comportement par défaut : 11 pour blastn, 3 pour les autres)
- f : seuil T de similitude des mots voisins pour étendre les hits
- X : seuil d'arrêt de l'extension avec brèche (quand le score de l'alignement tombe de X en dessous du meilleur score déjà observé). Par défaut 30 pour blastn, 0 pour tblastx, 15 pour les autres
- Z : seuil d'arrêt de l'extension avec brèche pour les alignements finaux (plus grand que X). Par défaut : 30 pour blastn, 0 pour tblastx, 25 pour les autres
- g : permettre un alignement contenant des gaps (F,T)

Il existe également un autre programme livré avec BLAST 2 : Netblast. Netblast est un client local appelé par la commande blastcl3 qui interagit directement avec les serveurs du NCBI. Il envoie la requête et les paramètres de recherche choisis (avec les mêmes options que blastall) par l'intermédiaire de la connexion à l'Internet. Il reçoit ensuite les résultats et les sauvegarde localement dans le format demandé [20]. Voir également le début de la référence de Netblast placé en annexe 4.

C. PSI-BLAST permet de trouver des relations biologiques faibles

PSI-BLAST prend une séquence de protéine et la compare à une base de donnée de protéines, en utilisant blastp. En prenant tous les alignements significatifs trouvés (l'utilisateur règle l'E-value adoptée comme seuil), le programme construit un alignement multiple. Il produit ensuite à partir de cet alignement un profil (un tableau poids-position[14], voir l'encadré ci-dessous) dont la longueur est identique à la requête initiale. Troisième étape, le programme compare le profil avec la base de donnée de protéines, en cherchant à nouveau des alignements locaux. A partir des séquences trouvées et en utilisant l'E-value seuil, il produit un nouvel alignement multiple, un profil affiné et peut relancer une nouvelle recherche. PSI-

14 En anglais *Position-Specific Scoring Matrix* ou PSSM

BLAST procède donc par itération, un nombre de fois déterminé par l'utilisateur ou jusqu'à la convergence (quand aucune nouvelle séquence similaire ne peut être trouvée) [5].

PSI-BLAST permet de découvrir de nombreuses séquences protéiques manquées par les autres méthodes (blastn, blastp...). Le programme est donc intéressant pour trouver des séquences divergentes de la requête.

> Comment construire un profil d'une collection de séquences ?[15]
>
> La première étape est de réaliser un alignement multiple des séquences. On prépare ensuite un tableau qui possède autant de lignes qu'il y a de caractères à considérer (4 lignes pour les 4 bases des séquences nucléotidiques, ou bien 21 lignes pour les acides aminés des séquences de protéines si l'on compte les brèches comme un caractère [3]). La tableau aura autant de colonnes qu'il y a de positions dans le signal que l'on souhaite représenter (ici on souhaite obtenir un profil de même longueur que la requête initiale). Le contenu du tableau va représenter le nombre de fois où un caractère est trouvé à la i^{ème} position.
>
> Par esprit de simplification, nous donnerons un exemple de tableau pour des nucléotides, mais il faut se rappeler que PSI-BLAST ne s'applique qu'à des séquences protéiques (pour lesquelles le tableau se construit de la même manière).
>
> ```
> Pos.  1 2 3 4 5 6 7 8 9 10
>
> A     6 9 8 5 2 1 4 5 7 4
> T     6 9 2 4 6 5 4 1 2 5
> C     7 1 7 5 8 9 7 5 3 3
> G     1 1 3 6 4 5 5 9 8 8
> ```
>
> Ce tableau montre le nombre d'occurrences de chacune des entités à chaque position de l'alignement multiple : par exemple à la première il y a 6 adénines, 6 thymines, 7 cytosines et 1 guanine pour un alignement de 20 séquences.
>
> Ce tableau est converti en tableau de fréquences. Chaque valeur devient la fréquence relative d'apparition d'un caractère donné à la position i du motif. C'est ce qu'on appelle le tableau-poids position (ou *Position Specific Scoring Matrix*, PSSM).
>
> ```
> Pos.  1     2     3     4     5     6     7     8     9     10
>
> A     0,3   0,45  0,4   0,25  0,1   0,05  0,2   0,25  0,35  0,2
> T     0,3   0,45  0,1   0,2   0,3   0,25  0,2   0,05  0,1   0,25
> C     0,35  0,05  0,35  0,25  0,4   0,45  0,35  0,25  0,15  0,15
> G     0,05  0,05  0,15  0,3   0,2   0,25  0,25  0,45  0,4   0,4
> ```

15 Sources : les articles sur PSI-BLAST ([3], [5], [8]), communication orale du 20 Octobre 2010 à l'ENS Ulm de Jacques van Helden (BiGRe, Université Libre de Bruxelles) sur la recherche de motifs, voir également la référence [9] page 241 pour une explication claire en français du tableau poids-position.

Partant de cette matrice, on peut extraire une séquence consensus possible en observant les plus fortes fréquences à chaque position :

Pos.	1	2	3	4	5	6	7	8	9	10
Seq.	C	A	A	C	T	C	C	G	A	T
Frq.	0,35	0,45	0,4	0,25	0,4	0,45	0,35	0,45	0,35	0,25

Je peux également produire une représentation logo :

Figure 10 : séquence logo réalisée à partir d'un tableau poids-position
Illustration Jacques van Helden *et al.*, *Theoretical and empirical quality assessment of transcription factor-binding motifs*, NAR 2010, 1-17 – Creative Commons

Cependant, PSI-BLAST utilise seulement le tableau poids-position. La fréquence d'apparition de la base dans cette matrice (PSSM) peut être vue comme la probabilité d'apparition de cette base. Partant de ce postulat, rechercher les occurrences du profil (que l'on utilise le consensus ou le tableau poids-position selon le cas de figure) sur les séquences de la banque interrogée ne présente pas de difficulté majeure [9].

Il faut noter que Lipman *et al.* considèrent comme une erreur d'attribuer à toutes les séquences de l'alignement multiple un poids égal et effectuent un calcul de fréquence plus compliqué ([3], page 3395). Selon eux, il n'y pas que le nombre de fois où un résidu est observé dans une colonne qui soit important. Il faut aussi prendre en compte le nombre de fois où il a été effectivement observé de manière indépendante. Le nombre d'observations indépendantes est différent du nombre de séquences présentes dans l'alignement multiple. En effet, 10 séquences identiques impliquent moins d'observations indépendantes que 10 séquences divergentes. (Pour des détails, voir [3]).

Attention cependant, PSI-BLAST augmente la quantité requise d'analyse des résultats de ses itérations par rapport aux autres programmes BLAST [5]. Certains auteurs n'hésitent pas à le décrire comme un des plus puissants programmes de recherche par similarité tout en rappelant qu'un œil particulièrement critique doit être appliqué aux résultats qu'il fournit [8].

En effet, lors d'une itération *n*, si une nouvelle séquence a été obtenue par similarité avec le profil qui sert de requête à l'itération en cours et que son score est suffisant pour l'incorporer

au profil utilisé à l'itération *n*+1, toutes les itérations suivantes rapporteront des séquences similaires à celle-ci [5]. Parfois, de nouvelles séquences nécessitent une corroboration avec d'autres données pour valider leur insertion dans la construction du profil *n*+1, notamment si leur E-value n'est pas très petite. Si aucune preuve ne vient corroborer le fait qu'il y a homologie entre cette séquence et le profil qui l'a généré alors il se peut que cette séquence n'ait aucun rapport biologique avec la requête initiale. La prendre en compte est une erreur. Or une erreur est facilement amplifiée par les itérations du programme, ce qui peut mener à des analyses biologiques complètement erronées.

> Recommandation : utiliser PSI-BLAST avec précaution

a) Évaluation de la méthode

Altschul et Koonin montrent un exemple où une corroboration est requise [5]. Ils utilisent une protéine non caractérisée (MJ0414 de *Methanococus jannaschii*) et lancent PSI-BLAST en choisissant un seuil tel que les séquences trouvées avec une E-value supérieure à 0,01 ne soient pas considérée comme significatives.

> A la date de l'article, en 1998, l'E-value est par défaut à 0,001, ce qui est conservateur – plus l'E-value est petite, plus les séquences retenues seront similaires[16].

La première étape (un blastp avec brèches) permet d'obtenir 4 séquences d'*Archea* et de bactéries avec une E-value comprise entre $1e^{-38}$ et $6e^{-77}$. La première utilisation du profil dérivé de ces 4 séquences permet de découvrir une ADN ligase II de levure avec une E-value de 0,005, ce qui n'est que modérément significatif. A l'itération suivante de nombreux alignements significatifs impliquent d'autres ADN ligases, ce qui est logique d'après les principes exposés plus hauts (rapatriement des séquences du fait de l'incorporation de l'ADN ligase II). En revanche, les auteurs se demandent s'il était bien raisonnable d'ajouter au profil la séquence qui possédait une E-value de 0,005. Ils considèrent qu'il est nécessaire d'établir une corroboration avec d'autres informations. Ils regardent alors l'alignement entre ce sujet et la requête initiale. Ils établissent alors qu'il y a effectivement des domaines caractéristiques d'ADN ligase dans la requête initiale. Ce n'est qu'après cette étape que les auteurs s'autorisent à considérer la prise en compte de cette séquence, ce qui souligne la précaution avec laquelle ceux-ci utilisent le programme.

16 En 2002, le seuil par défaut était ramené plus grand à une E-value de 0,005, donc moins conservatif que les premiers réglages [8]. Le seuil est toujours réglé à cette valeur sur l'interface web du NCBI au 20 Octobre 2010.

b) *Utilisation en ligne de commande*

L'application blastpgp permet de réaliser à la fois des PSI-BLAST et des PHI-BLAST.

Un exemple d'utilisation de PSI-BLAST est : `blastpgp -i my_kinase.txt -d refseq_protein -j 6 -o my_kinase.out`.

> Quelques paramètres de blastpgp pour PSI-BLAST [19]
>
> -j est le nombre maximal d'itérations (par défaut 1).
> -h est le seuil de l'E-value pour inclure les séquences dans le tableau poids-position. (par défaut 0.001).
> - C permet la sauvegarde d'un tableau poids-position. "-u 1" ordonne à blastpgp de le sauvegarder au format texte. (Dans ce cas l'option -J T est également requise : `blastpgp -i query -d refseq_protein -C chk.asn -j 2 -u 1 – J T > /dev/null`). La matrice enregistrée est celle utilisée à la dernière itération.
> - R permet de poursuivre des itérations depuis un fichier sauvegardé.
> -d comme à l'habitude spécifie la banque interrogée.
> - B fournit un moyen de faire démarrer PSI-BLAST depuis un alignement multiple calculé à l'extérieur de BLAST. Cet alignement doit être dans un format dérivé du format produit par Clustal. L'alignement multiple doit inclure la séquence requête.

c) *Note sur PSI-TBLASTN*

PSI-BLASTN est une variante de tblastn et recherche un requête (en protéine) contre une banque de nucléotides traduite dans toutes les phases de lectures, de la même manière que tblastn. Une itération de PSI-BLAST est cependant effectuée pour créer une matrice qui va servir à évaluer le score des touches et des alignements, au lieu d'utiliser une matrice PAM ou BLOSUM.

D. PHI-BLAST : la stratégie pour ancrer la similarité avec un motif

Un article publié en 1998 par Zhang, Lipman *et al.* introduit un nouvel algorithme dans la galaxie [6]. Il s'agit de PHI-BLAST, une application qui recherche des séquences de protéines par similarité à partir d'une ancre. L'ancre est un motif qui est donné en même temps que la séquence requête et doit être contenu au moins une fois dans celle-ci [6]. Les auteurs rapportent que des méthodes précédentes avaient été mises au point pour rechercher les motifs *dans* les résultats d'un BLAST standard. En revanche, le nouvel algorithme est une combinaison hybride de la recherche de motif avec la recherche de séquences ayant une

similarité significative avec la requête. Pour chaque correspondance entre une occurrence du motif dans la requête et une occurrence dans une séquence de la banque ou base interrogée, PHI-BLAST construit un alignement local avec brèches de score le plus élevé possible incluant la correspondance. Comme à l'habitude, tous les alignements sont triés par scores et évalués statistiquement [6]. Le motif est une séquence de résidus définie par une expression régulière, avec des jokers et des espaces, par exemple [GS]-x-S-M-x-P-[AT]-[LF].

Ce programme peut permettre de répondre à la question suivante : est-ce qu'un motif vu dans une protéine d'intérêt est nécessaire pour (ou simplement impliqué dans) la fonction assurée par la protéine, ou bien y est-il inséré par hasard ?

La définition du score utilisé pour évaluer la plausibilité du résultat est différente des programmes précédemment présentés. Les auteurs décomposent le score en trois parties : le score de la touche correspondant au motif (S_0), les scores des extensions de part et d'autres de la touche (S_1 et S_2). Parce qu'ils considèrent que tous les alignements avec le motif sont de plausibilités biologiques égales, le score s0 n'est pas pris en compte. Seul est pris en compte le score réduit $S' = S_1 + S_2$.

Pour de grands scores x, le distribution de S' est de la forme $P(S' \geqslant x) \sim C(\lambda . x + 1) e^{-\lambda . x}$. C est estimé à 0,6 pour une BLOSUM 62 (simulation aléatoire avec les fréquences de Robinson et Robinson [6]). Des tables de λ sont disponibles pour de nombreuses matrices de substitution et de coûts de brèche [6]. Il en vient que le nombre d'alignements produits par hasard par PHI-BLAST avec un score réduit x est donné par l'équation suivante : $E(S' \geqslant x) \sim C(\lambda . x + 1) e^{-\lambda . x}$.

> Il est possible de demander à PHI-BLAST de donner en sortie le tableau-poids position du motif recherché. Ce tableau peut ensuite être donné à PSI-BLAST. Il existe également la possibilité de commencer par rechercher des motifs sur PROSITE, puis de les utiliser dans une recherche PHI-BLAST.

a) Utilisation en ligne de commande

En ligne de commande, l'utilisation de PHI-BLAST est de la forme `blastpgp -i query -k pat -p patseedp -d refseq_protein -o refp_pat.out` où -i est suivi par le fichier contenant le requête au format FASTA, -k est suivi par le fichier de motif au format PROSITE et où `patseedp` indique le mode d'utilisation de blastpgp.

Il existe un deuxième mode d'utilisation de PHI-BLAST dont l'usage est important quand le motif spécifié apparaît plusieurs fois dans la requête. L'utilisateur peut restreindre la recherche d'alignements locaux à une sous-partie des occurrences en utilisant les lignes *HI*. Ce mode est appelé par la commande `blastpgp -i -k -p seedp`.

```
ID    ER_TARGET; PATTERN.
PA  [KRHQSA]-[DENQ]-E-L>.
HI (19 22)
HI (201 204)
```

Dans cet exemple de fichier de motif, les lignes HI spécifient deux occurrences du motif dans la requête qui vont être utilisées pour les alignements, des positions 19 à 22 puis des positions 201 à 204[17]. L'interface web du NCBI ne permet que d'utiliser PHI-BLAST en mode « patseedp ».

Il est possible de réaliser un PHI-PSI-BLAST. Le premier tour de recherche fait appel à PHI-BLAST, les autres à PSI-BLAST. PSI-BLAST est alors invoqué avec l'option -j à la suite des modes décrits ci-dessus : `blastpgp -i -k -p patseedp -j`.

Toutes ces commandes peuvent être retrouvées dans l'aide en ligne du NCBI à l'adresse en référence [19][18].

b) Évaluation des performances de la méthode

L'efficacité de PHI-BLAST est testée par les auteurs. Les ARNt-nucléotidyletransférases d'archée (Cca) sont une famille de polymérases d'acides nucléiques. Dans une recherche standard aucune similarité ne peut être trouvée. Cependant, elles possèdent un motif conservé avec deux résidus Aspartate qui ressemble au site catalytique de nombreuses autres polymérases. Quand ce motif est donné à PHI-BLAST conjointement à la Cca d'une archée certaines séquences sont obtenues : une protéine hypothétique (AF0299) d'une autre archée annotée et prédite comme nucléotidyletransférase (E-value de 0,061) ; une protéine de *Enterococcus faecalis* validée expérimentalement comme étant une streptomycine-3'-adénylyltransférase (E-value 0,13).

On observe que l'E-value de ces séquences est peu comparable à celles obtenues lors d'une recherche simple de protéines plus répandues, mais PHI-BLAST permet de trouver des

17 En d'autres termes, les lignes HI du fichier de motif ordonnent à PHI-BLAST de se concentrer sur ces occurrences spécifiques pendant la recherche.

18 Remarque. La dernière mise à jour de cette page de manuel au NCBI est récente : elle date du 20 octobre 2010. Cette dernière version est plus claire que les innombrables versions recopiées sur d'autres serveurs.

similarités, témoignant de la puissance de ce programme. Cela n'empêche que les résultats doivent être corroborées par d'autres informations (ici, une validation expérimentale).

> De la même manière que PSI-BLAST, les résultats de PHI-BLAST devraient être analysés avec attention. C'est la contre-partie de la puissance de ces logiciels.

E. PSI-BLAST inversé : pour tester l'existence d'un motif connu

Le PSI-BLAST inversé (rpsblast) interroge une base de données de motifs ou tableaux poids-position (PSSM) à partir d'une requête constituée par une ou plusieurs séquences de protéines [20] pour identifier ceux avec lesquels la requête présente des similitudes. rpsblast utilise un algorithme proche de celui de autres programmes BLAST (recherche d'une simple ou double touche puis réalisation d'une extension sans brèche. Si le score de cet alignement est suffisamment élevé, une extension avec brèches est réalisée et les alignements avec une faible E-value sont montrés dans les résultats d'exécution). Cela permet de savoir si notre requête contient des motifs connus.

III - Le temps de la réécriture : BLAST en C++

BLAST est un ensemble de programmes qui n'ont pas vocation à rester statiques. Ceux-ci ont fait l'objet de continuelles et graduelles améliorations. En Avril 2008, une publication fait la promotion de la nouvelle interface graphique pour les applications Web [16], mise en ligne un an plus tôt (le 16 avril 2007 [14]). La version BLAST 2.2.21 sortie le 28 Juillet 2009 [14] inclut de nouvelles applications en ligne de commandes, dénommées BLAST+, complètement réécrites à partir de librairies modernes. Enfin, la documentation en ligne au NCBI a été mise à jour pendant l'écriture de cette synthèse, en Octobre 2010. Ces quelques éléments prouvent que BLAST est loin d'être un paquet logiciel moribond des années 1990, abandonné par ses développeurs. Ce chapitre se propose de passer en revue les améliorations fournies par BLAST à partir de la version 2.2.21.

A. Motivations amenant à la réécriture du code

Une des motivations de la réécriture du code avec des librairies modernes[19] est de pouvoir augmenter sa rapidité d'exécution. La décomposition du problème en parties plus simples est un moyen employé pour y parvenir, comme nous le verrons dans le paragraphe dédié aux améliorations informatiques. L'augmentation ininterrompue de la taille des banques et bases de données (et de la taille des séquences dans celles-ci) est de nature à contrecarrer les méthodes rapides développées dans les années 1990 (notamment, l'exploration partielle du graphe en utilisant un seuil X). Le nombre d'utilisateurs toujours plus important amène à optimiser la répartition de la charge de calcul sur les serveurs des centres mettant BLAST à disposition[20]. Enfin, l'utilisateur sera satisfait d'avoir un programme qui rapatrie des séquences en quelques secondes, même quand de très nombreuses personnes interrogent le moteur en même temps. En 2010 il y a environ 350 000 interrogations de BLAST sur les serveurs du NCBI par jour ouvrable (Thomas Madden, communication personnelle), à comparer avec les valeurs antérieures pour 2002 et 2004 ([12], figure 11).

19 Officiellement sur une suggestion de David Lipman et Jim Ostell.

20 Dans cette étude, nous nous contenterons de traiter le cas publié des serveurs du NCBI. Il sort du cadre de cette synthèse d'enquêter sur les ressources allouées à BLAST sur d'autres serveurs, comme ceux de l'EMBL. Néanmoins, si vous possédez de telles informations, vous pouvez les transmettre à l'auteur en utilisant l'adresse de contact donnée au début de ce travail.

Une autre préoccupation est de garantir à long terme l'architecture modulaire de ces programmes pour permettre leur évolution (ajouts et modifications de fonctions tout en conservant un code clair et manipulable).

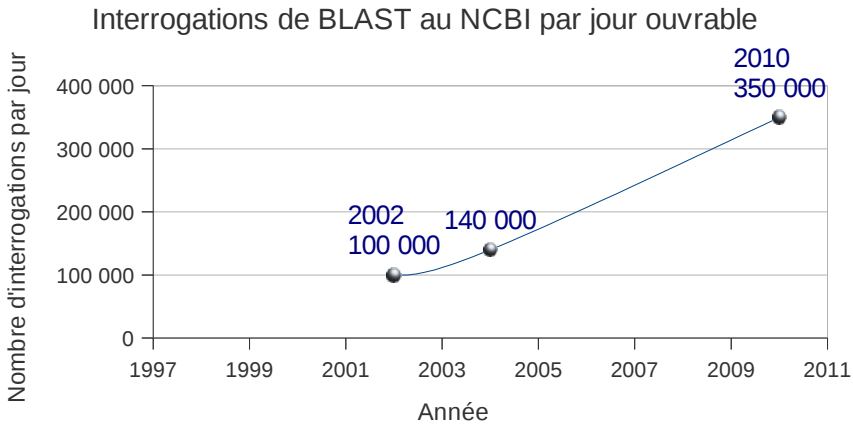


Figure 11 : évolution des interrogations de BLAST sur les serveurs du NCBI entre 2001 et 2010

B. Correspondance avec les anciens programmes

L'idée des développeurs de BLAST est de rendre plus agréable l'expérience utilisateur (mais pas seulement). Cette volonté transparait en deux points. Premièrement, le concept de tâche a été renforcé [17]. Pour chaque tâche le biologiste dispose d'un programme bien identifié. Choisir le bon programme est déjà une façon d'adopter des paramètres optimisés pour la tâche. Par exemple, MEGABLAST et BLASTN sont tous les deux des programmes de recherche de nucléotides contre une banque de nucléotides mais choisir MEGABLAST permet d'optimiser la recherche pour des séquences faiblement divergentes. En effet, MEGABLAST inclut par défaut un mot initial plus grand (28 caractères) et des pénalités de brèches linaires. Deuxième point, le biologiste dispose maintenant d'arguments explicites pour les lignes de commandes (par exemple, « word_size » à la place de « w » dans blastall [17]). La correspondance avec les anciens programmes est présentée en figure 12.

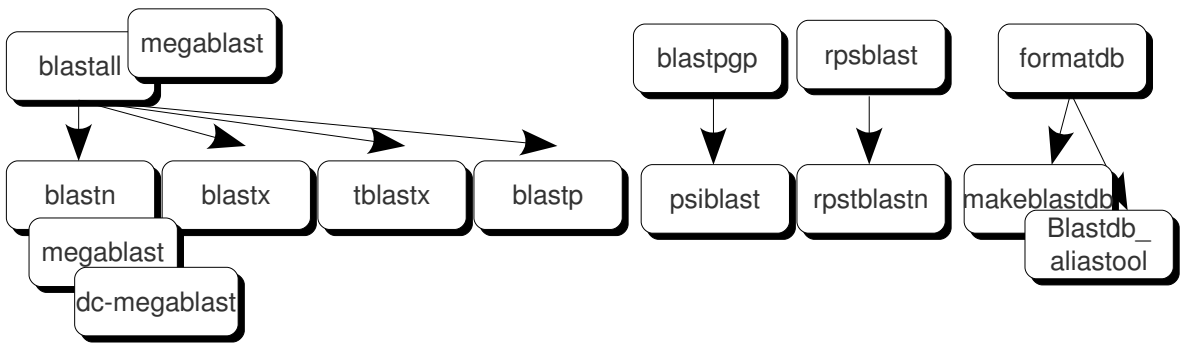


Figure 12 : correspondance entre BLAST 2 et BLAST+. Les fonctionnalités offertes par les applications BLAST+ sont organisées par types de programmes.

A la date d'écriture de ce document, l'ensemble des anciennes applications blastall, blastpgp, rpsblast, bl2seq sont toujours distribuées par le NCBI sous le nom de *Legacy Blast* (en fait, le cœur de *Legacy* a tout de même évolué). Le NCBI propose un guide de sélection du programme, document incontournable pour identifier le programme qui correspond à une tâche et plus généralement pour se former à BLAST (consulter [20] et [22]). Des informations sur le MEGABLAST discontinu, non détaillé dans la présente synthèse mais proche parent de MEGABLAST, y sont notamment figurées.

Utilisation en ligne de commande

Les auteurs de BLAST+ permettent aux utilisateurs historiques de BLAST 2 de continuer à utiliser les instructions qu'ils connaissent. Il suffit de les exécuter dans le cadre d'un script de compatibilité (legacy_blast.pl). Ce script convertit les instructions BLAST 2 en instructions BLAST+ en appelant le programme adéquat [18].

> Les primo-utilisateurs de BLAST devraient directement apprendre à se servir des nouvelles commandes explicites, intégralement détaillées dans les tables supplémentaires (http://www.biomedcentral.com/content/supplementary/1471-2105-10-421-s1.pdf) de l'article introduisant BLAST+ donné en réf. [17].

Le tableau suivant n'a pas la prétention de donner les clés d'utilisation de BLAST+. Il se contente de comparer pour un exemple simple les commandes à exécuter pour réaliser une tâche similaire avec BLAST 2 et BLAST+.

Tableau 1 : comparaison des commandes pour rechercher par similarité des séquences nucléotidiques

BLAST < 2.2.21 ou *Legacy* BLAST	`blastall -p blastn -i seq.fasta -d mabanque -o resultat -W 11 -X 30 -Z 100 -G 5 -E 2 -r 2 -q -3`
BLAST+ (> 2.2.21)	`legacy-blast.pl blastall -p blastn -i seq.fasta -d mabanque -o resultat -W 11 -f 13 -X 30 -Z 100 -G 5 -E 2 -r 2 -q -3 -- path /opt/blast/bin` `blastn -query seq.fasta -db mabanque -out resultat -wordsize 11 -xdrop_gap 30 -xdrop_gap 100 -gapopen 5 -gapextend 2 -reward 2 -penalty -3`

A noter également : l'option « -f » (le score minimal pour ajouter un mot à la liste de mots voisins pour les protéines) de blastall devient « threshold » dans blastp. L'option « -M » (par

exemple `-M BLOSUM62`) devient « matrix » (l'usage devient donc : `-matrix BLOSUM62`). Enfin, le programme netblast devient obsolète avec l'introduction de l'option « remote » pour effectuer la recherche directement avec le moteur compilé sur les serveurs propres du NCBI, et utiliser leurs bases de données.

C. Améliorations informatiques

A côté de l'ajout de petites fonctionnalités pratiques telles que la possibilité de renseigner directement le numéro de la fiche GenBank dans laquelle se trouve la séquence requête (au lieu de donner celle-ci au format FASTA), BLAST+ fait l'objet de changements majeurs au niveau de l'architecture des programmes.

1) Optimisation de la gestion des processeurs et de la mémoire

Les auteurs ont voulu réduire le temps de calcul par le processeur et l'empreinte mémoire. Une façon d'améliorer les performances est de contraindre la table des mots voisins et la table de diagonales décrivant les extensions des touches originelles au cache mémoire L2 [17]. Une estimation de la taille en octet de ces deux éléments pour une requête de longueur N est : $528{,}384 + 8N$. Pour une requête de taille N = 50, on approche du méga-octet, à comparer avec la taille du cache mémoire L2 pour un processeur Intel Xeon : de 0,5 à 4 Mo. Ainsi, pour des requêtes beaucoup plus grandes (par exemple un contig), le cache est dépassé. Pour résoudre ce problème, BLAST+ réalise un découpage de la requête en petites parties chevauchantes. Pour la phase de recherche dans la base. BLAST fusionne ensuite les résultats et aligne la requête entière pendant la phase finale d'enregistrement de l'alignement. L'estimation de la mémoire occupée devient alors $266{,}240 + 8N$ [17]. Dans leurs tests d'efficacité de la méthode, les auteurs rapportent que le temps d'exécution est réduit de 2% pour de grandes requêtes (8000 nucléotides).

2) Séparation entre algorithme et traitement des données

La conception actuelle de BLAST vise à séparer la partie algorithmique de BLAST des modules qui rapportent les séquences depuis les banques ou bases interrogées [17]. Ceci pour permettre au nouveau code de BLAST d'être utilisé à la fois dans les environnements en C et

en C++, tout en interrogeant le même cœur de programme[21] [17]. Exécuter BLAST dans un environnement C++ est devenu possible depuis le développement d'une nouvelle boite à outils (*toolkit*) C++ au NCBI, contenant plusieurs modules de codes, dont l'interface de programmation (*Application Programming Interface* ou API) de BLAST [21].

Le code totalement réécrit des nouvelles versions de BLAST est donc scindé en deux catégories distinctes. Il existe d'une part le code au cœur de BLAST écrit en *vanilla C* et qui n'utilise aucune boîte à outils C ou C++ du NCBI. On trouve d'autre part le code de l'interface de programmation (*Application Programming Interface* ou API) qui est écrit en C++ qui exploite les avantages de la nouvelle boîte à outils C++. La raison qui a amené à écrire le cœur en *vanilla C* est que le même code pouvait être utilisé avec l'ancien *toolkit* C (et remplacer ainsi le code de 1997). Il devient également possible aux chercheurs intéressés par le développement de l'algorithme de travailler avec le cœur de BLAST indépendamment de toute boîte à outils. Il est possible d'isoler le développement de chacune des deux parties, de façon à ce que les améliorations algorithmiques soient transparentes au programmeur de l'interface de programmation. Cependant, étant donné que BLAST est continuellement en développement et que nombre de développements consistent à ajouter de nouvelles fonctions, les auteurs rapportent cependant qu'il n'est pas toujours possible ou désirable d'isoler le programmeur API de ces changements [21]. Quoiqu'il en soit l'API ne doit pas « réfléchir », son objectif est de présenter des résultats et n'a pas vocation à inclure de l'algorithmie[22].

> Si à partir de la version 2.2.21 BLAST2 (basé sur le *toolkit* C) et BLAST+ (partiellement basé sur le *toolkit* C++) partagent un même cœur on peut se demander quelle est la différence entre les deux programmes. La différence se trouve dans les modules de traitement : le lecteur de base de donnée, le module de formatage de sortie, le module de césure de la requête. L'intérêt de posséder des briques simples séparées du cœur de l'algorithme (qui se charge de passer les informations d'un module à l'autre) est de pouvoir les améliorer, les modifier ou les remplacer sans conséquences sur l'architecture du traitement. Avec les anciennes librairies C, on utilise un vieux lecteur de base de données (*readdb*), mais un nouveau a été écrit avec les nouvelles librairies (*CseqDB*), implémentant des fonctionnalités exclusives. Le formateur de sortie a été totalement réécrit. Enfin, la césure des requêtes n'est possible qu'avec l'API C++. (Thomas Madden, communication personnelle et références [17], [21]). Il vaut donc

21 BLAST fonctionne avec ISO C99 (ce qui permet l'utilisation dans les deux environnement sus-mentionnés) et utilise un ADT (*Abstract Data Type*), BlastSeqSrc, pour rapatrier les séquences sujets qui vont être traitées par le code au cœur de BLAST [17].

22 L'auteur de ces lignes remercie T. Lefèvre, ingénieur en informatique, pour ses remarques à ce sujet.

mieux utiliser le dernier BLAST+, d'autant plus que les arguments en ligne de commande sont désormais explicites, comme on l'a vu.

3) Répartition de la charge sur le serveur du NCBI

Les informations disponibles dans la littérature indiquent le doublement régulier de la capacité de traitement. Ainsi en 2004 200 processeurs (répartis sur des machines à double cœur) étaient disponibles [12]. Deux ans auparavant, 100 processeurs étaient disponibles [12]. Le système permettait aussi de savoir sur machines une base de données précise avait été chargée suite à une requête, de façon à ce que les requêtes suivantes utilisant la même base soient dirigée sur ces machines (où la base est probablement toujours en mémoire).

A la date de l'article en référence [12] le système conservait les requêtes, les résultats et des statistiques variées sur une paire de machines sous Microsoft SQL Server 2000. Dans l'article, il est mentionné que la puissance devait doubler encore à l'horizon des deux ans à venir (soit, à partir de 2004, à l'horizon 2006). En 2010, on peut légitimement supposer que le serveur repose au moins sur 500 processeurs répartis sur des machines à double cœur.

D. Autres améliorations

1) Le masquage de la base de données

Les régions de faible complexité et les régions intergéniques répétées sont de natures à provoquer de nombreuses touches sans intérêt biologique. BLAST permet de masquer ce types de régions dans la requête et dans la base de donnée interrogée. Deux modes de masquage existent : le mode sévère et le mode léger (voir la figure 13). Le mode sévère remplace ces régions par des caractères joker (N pour les nucléotides, X pour les protéines) pour toutes les phases de la recherche. Le masquage léger fait que les portions masquées de la requête sont inutilisables pendant la phase de recherche des touches initiales, mais sont disponibles une fois que les touches ont été enregistrées pour les différentes phases d'extension. Le masquage de la base de donnée ne peut être réalisé qu'en mode de masquage léger [17].

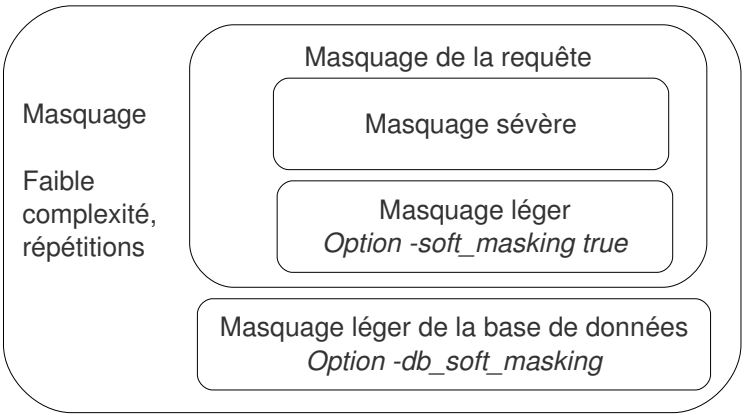


Figure 13 : les différents modes de masquages fournis par BLAST+.

Originellement, il n'était possible d'appliquer le masquage qu'avec un programme d'analyse de la séquence requête, utilisé en amont de BLAST[23].

2) Primer BLAST

Primer-BLAST est une application rendue disponible en Juillet 2008. Elle permet de rechercher des amorces pour une réaction de polymérisation en chaine. Le programme combine la conception d'oligonucléotides (avec le logiciel Primer 3) et un test de spécificité par BLAST.

°=°=°

23 Une variante de BLAST disponible à l'adresse http://www.repeatmasker.org/RMBlast.html propose l'intégration d'un tel programme amont, RepeatMasker avec le BLAST de NCBI.

Conclusions et perspectives

Cette synthèse a mis en évidence l'impact des algorithmes heuristiques de BLAST. Les articles de 1990 et 1997 ont été parmi les articles les plus cités dans la communauté scientifique. Chaque jour le serveur du NCBI (auquel il faudrait ajouter celui de l'EMBL et de multiples autres instituts) reçoit 350 000 requêtes. Enfin, la recherche par similarité permise par ces programmes est quasiment systématiquement employée dans les routines d'études de séquences, notamment lors de l'annotation de génomes.

Pour un chercheur non génomicien BLAST est une bonne façon de découvrir les facettes d'un gène. Il suffit pour cela de bien identifier la tâche à effectuer (comme rechercher des séquences nucléiques similaires à une requête nucléique) puis se servir du récent guide en ligne pour choisir le programme approprié. Cette façon de procéder devrait amener plus rapidement (et de manière transparente) les utilisateurs novices de BLAST à optimiser les paramètres de recherche. Pour les chercheurs qui désirent faire une analyse plus fine, il reste possible de faire de très nombreux choix (souvent au cas par cas) en utilisant les options du programme et en utilisant des routines plus complexes, telle que la procédure de PHI-PSI-BLAST, ou celle de PSI-BLAST inversé. Ainsi, BLAST peut être utilisé pour rechercher de grandes séquences dans des bases de données mais aussi de courts motifs représentant des domaines ou des signatures particulières. Dans ce dernier cas il faut rapporter l'existence d'autres algorithmes, reposant sur des modèles de Markov cachés et utilisant des jeux d'entrainement. Les résultats de ces algorithmes gagneraient à être comparés aux résultats de routines BLAST intégrant des tableaux poids-position (PSSM) habilement déterminés, ce qui sort du cadre de cette étude[24].

BLAST est un programme activement maintenu depuis 20 ans. Ainsi, la version 2.2.21 a été publiée en Juillet 2009, la version suivante en Octobre de la même année, la suivante en Mars 2010 et la 2.2.24 actuelle en Août 2010. C'est un logiciel libre. Il a donc été porté sur de nombreuses plate-formes (*cf.* tableau 2).

24 L'auteur étant désireux d'avoir plus de détails sur l'efficacité (et les conditions) des approches BLAST comparées aux modèles de Markov pour la recherche de courts motifs, toutes les remarques et suggestions de lectures émises *via* l'adresse de contact seront les bienvenues.

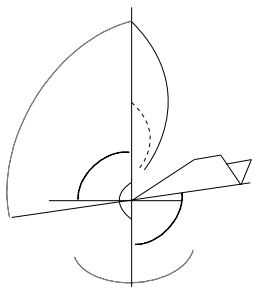
Tableau 2 : test de quelques bases de données pour la possession d'un outil BLAST

	ensembl	KEGG	DDBJ	GOLD	CyanoList	FlyBase	ParameciumDb	ACLAME	Tigr fams	Uniprot
BLAST ?	Oui	Oui Ĥ	Oui	Non	Oui	Oui	Oui	Oui	Non Ĥ	Oui

Ĥ : présence d'un programme basé sur les modèles de Markov

Son code source étant libre, de nombreuses modifications ont été réalisées. Cela montre à quel point la voie est libre pour le développement de nouvelles fonctions, l'amélioration des méthodes statistiques embarquées et l'optimisation des ressources informatiques nécessaires, notamment par la parallélisation du traitement. On citera AB-BLAST (http://www.advbiocomp.com/blast.html) remplaçant du défunt Wu-BLAST[25], mpiBLAST (http://www.mpiblast.org/) et Hi-per BLAST. L'étude des différences (à la fois en algorithmie et en efficacité de réponse à une question biologique) entre ces derniers paquets logiciels et celui développé au NCBI serait une intéressante poursuite de ce travail.

Pour finir, cette synthèse a permis de constater que les auteurs de BLAST au NCBI ont adopté une nouvelle architecture modulaire. De nouveaux modules peuvent être développés sans toucher à l'algorithme de BLAST. On peut supposer que cette façon claire de structurer le programme amènera encore de nouvelles et nombreuses contributions à la galaxie BLAST.



25 Wu-Blast est encore disponible sur l'interface Mobyle de l'Institut Pasteur.

Références

1 : Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers et David Lipman (1990), Basic Local Alignment Search Tool, *Journal of Molecular Biology*, **215**, 3, 403-410.

2 : Samuel Karlin et Stephen Altschul (1990), Methods for assessing the statistical signifiance of molecular sequence feature by using general scoring schemes, *Proceedings of the National Academy of Sciences of the United States of America*, **87**, 6, 2264-2268.

3 : Stephen F. Alschul, Thomas L. Madden, Alejandro A. Schäfer, Junghui Zhang, Zheng Zhang, Webb Miller et David J. Lipman (1997), Gapped BLAST and PSI-BLAST : a new generation of protein database search programs, *Nucleic Acids Research*, **25**, 17, 3389-3402.

4 : NCBI, NLM, NIH (1997), What's new Archive, *http://www.ncbi.nlm.nih.gov/About/whatsnew97.html*

5 : Stephen F. Altschul, Eugène V. Koonin (1998), Iterated profile searches with PSI-BLAST - a tool for discovery in protein databases, *Trends in Biochemical Sciences*, **23**, 11, 444-447.

6 : Zheng Zhang, Alejandro A. Schäffer, Webb Miller, Thomas L. Madden, David J. Lipman, Eugene V. Koonin et Stephen F. Altschul (1998), Protein sequence similarity searches using patterns as seeds, *Nucleic Acids Research*, **26**, 17, 3986-3990.

7 : Zheng Zhang, Scott Schwartz, Lukas Wagner et Webb Miller (2000), A Greedy Algorithm for Aligning DNA Sequences, *Journal of Computationel Biology*, **7**, 1/2, 203-214.

8 : David T. Jones, Mark B. Swindells (2002), Getting the most from PSI-BLAST, *Trends in Biochemical Sciences*, **27**, 3, 161-164.

9 : Claudine Médigue, Stéphanie Bocs, Laurent Labarre, Catherine Mathé, David Vallenet (2002), L'annotation in silico des séquences génomiques, *Médecine/Sciences*, , 18, 237-250.

10 : Jo McEntyre et Jim Ostell (2002-), The NCBI Handbook, *http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=handbook*

11 : Science Watch® (2003), Twenty Years of Citation Superstars, *http://archive.sciencewatch.com/index.html*

12 : Scott McGinnis, Thomas L. Madden (2004), BLAST : at the core of a powerful and diverse set of sequence analysis tools, *Nucleic Acids Research*, **32**, numéro du serveur web, W20-W25.

13 : E. Michael Gertz (2005), BLAST Scoring Parameters, *nebc.nerc.ac.uk/bioinformatics/documentation/blast/scoring.pdf.gz*

14 : National Library of Medicine, National Health Institutes of Health (2005-10), Page des nouveautés concernant BLAST, *http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?CMD=Web&PAGE_TYPE=BlastNews*

15 : Tao Tao (2006), Program Parameters for blastall, *http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/blastall/*

16 : Mark Johnson, Irena Zaretskaya, Yan Raytselis, Yuri Merezhuk, Scott McGinnis, Thomas Madden (2008), NCBI BLAST: a better web interface, *Nucleic Acids Research*, **36**, numéro du serveur web, W5-W9.

17 : Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, Thomas Madden (2009), BLAST+: architecture and applications, *BMC Bioinformatics*, **10**, 421, .

18 : Christiam Camacho, Thomas Madden, George Coulouris, Vahral Avagyan, Ning Ma, Tao Tao, Riche Agarwala (2009), Manuel de l'utilisateur des applications BLAST en ligne de commande (BLAST Command Line Applications User Manual), *http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=helpblast&part=CmdLineAppsManual*

19 : Tao Tao (2010), Aide de PSI et PHI-BLAST, *http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/blastpgp.html*

20 : NCBI (2010), BLAST documents - a personal collection (Dr. Tao), *http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/*

21 : Thomas Madden, Jason Papadopoulos, Christiam Camacho, George Coulouris et Kevin Bealer (créé en 2006, m.a.j. en 2010), BLAST API - référence du NCBI pour le kit C++, *http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=toolkit&part=ch_blast*

22 : National Library of Medicine, National Institutes of Health (version en date du 16/10/2010), Guide de sélection du programme BLAST (Blast Program Selection Guide), *http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=ProgSelectionGuide*

Annexe 1 : Shigella phage SP18, complete genome (GenBank GQ981382)

```
LOCUS       GQ981382              170605 bp    DNA     linear   PHG 12-OCT-2010
DEFINITION  Shigella phage SP18, complete genome.
ACCESSION   GQ981382
SOURCE      Shigella phage SP18
REFERENCE   1  (bases 1 to 170605)
  AUTHORS   Kim,K.H., Chang,H.W., Nam,Y.D., Roh,S.W. and Bae,J.W.
  TITLE     Phenotypic characterization and genomic analysis of the Shigella
            sonnei bacteriophage SP18
  JOURNAL   J. Microbiol. 48 (2), 213-222 (2010)
   PUBMED   20437154
REFERENCE   2  (bases 1 to 170605)
  AUTHORS   Kim,K.-H., Chang,H.-W. and Bae,J.-W.
  TITLE     Direct Submission
  JOURNAL   Submitted (21-SEP-2009) Department of Biology, Kyung Hee
            University, HoeGi-Dong 1, DongDaeMun-Gu, Seoul 130-701, Republic of
            Korea
FEATURES             Location/Qualifiers
     source          1..170605
                     /organism="Shigella phage SP18"
                     /mol_type="genomic DNA"
                     /host="Shigella sonnei"
                     /db_xref="taxon:645664"
     gene            complement(2..2236)
                     /gene="rIIA"
                     /locus_tag="SP18gp001"
     CDS             complement(2..2236)
                     /gene="rIIA"
                     /locus_tag="SP18gp001"
                     /note="similar to rIIA protector from prophage-induced
                     early lysis of Enterobacteria phage JS98"
                     /codon_start=1
                     /transl_table=11
                     /product="rIIA protector from prophage-induced early
                     lysis"
                     /protein_id="ADO19343.1"
                     /db_xref="GI:308205944"
                     /translation="MILETEKEAILGNGAKASAFTIQASPKVFKILTSDLYTNKVRAV
                     VRELITNMIDAHMLNGCQDKFIVQAPGELDPRFVCRDFGPGMSDFQIRGDENTPGLYN
                     SFFASSKAESNDFIGGFGLGSKSPFSYTETFSLISYHDGQVNGYVAYMDGDGPQIKPT
                     FSEPMKPGDRTGIEVVVPVDSNDFGKFRYEIAYIMRPFAGLAEVQGAGEIEYFPEFDD
                     KARNVETKFTKSQLTFNQMYEMFELDKDLINAGVVYDIYSTTRLKRLRNSSSNTNTAS
                     LTSMFGIHNDTLNIVIDDAKGRIAMIRGISSMLYDTSAKAKKLKASGKVPTVQGSSLL
                     FVDPTSEIQTRLMVQLQERFEGDTVNIYRTSELTALVKDWIPVADPKARVKSKTPSLL
                     RWTKTNGKWQSVEEFTCAADAEEIDGYAVFINRSSISPLNQEYGTTTFGTQTLCKLAN
                     LLEINEFCVIRPQLQKKVTKLAQYDCLIEAAVNRYVELIDAVDADQYIAASSRSRYYI
                     PILSEYAELNFMFKYFYAKDVQKDVNTDYAELMQFNKFFAYNAYNSGTVDKTMQETLV
                     LCNKIYDDLTKNASVTSDKMVFEFEKNHPIVSHFLYNRNAMSTEQVQNVAQIMKAVEA
                     ANKE"
(...)
ORIGIN
        1 attattcctt attggccgct tccacggcct tcatgatttg agcaacgttc tgcacttgtt
       61 cagttgacat ggcgttacgg ttatataaaa agtgtgaaac aattgggtga ttcttttcaa
          ...
```

http://www.ncbi.nlm.nih.gov/nuccore/GQ981382.1

Un gène a été identifié sur le brin complémentaire de la position 2 à la position 2236

La fiche se poursuit par de nombreuses autres annotations similaires

Séquence complète du génome (extrait)

Annexe 2 : communications de Dr. T. Madden

Le 12 octobre 2010
Guillaume Fernandez : The next step is the gapped extension using dynamic programming. I understand that the extension goes on while the score of the alignement drops no more than Xg (NAR paper) belows the best already-observed score.
I [understand] that Xg would be controlled by command line argument -X ! I am right ?
Default -X are 30 for blastn, 20 for megablast, 15 for all others. Not used by tblastx.
For me at this step, everything is over : there is only to extract the better alignement (as usual it is the result of dynamic programming).

Thomas Madden (NCBI, NLM, NIH) : Yes, -X controls the "initial" gapped alignment. This gapped alignment only calculates the score and extent of the alignment. The actual traceback (including indels) is not found. We store this information and rank the matches by the score (or it's expect value). We may eliminate matches that overlap with a higher-scoring match that has already been found. So, this does not really provide you with the full alignment, depending upon what you need.

G.F. : Something is controlled by parameter -Z... ("X dropoff value for final [dynamic programming] gapped alignment in bits"). Default -Z are 50 for blastn, 50 for megablast, and 25 for all others. Maybe I've missed something in 1997 BLAST paper. More likely we have a comprehension difficulty about the path graph explored and how BLAST2 makes it.

T.M. : For the final gapped alignment of the alignments to be presented to the user BLAST uses a large Xg value and actually stores the traceback (indels). This is the parameter -Z. Below is an excerpt from page 3393 of the 1997 article that mentions this, though I admit it's easy to overlook. The "Overall design" section of the BLAST+ article I mentioned also describes this.

« For any alignment actually reported, a gapped extension that records ‘traceback’ information (25) needs to be executed. To increase BLAST’s accuracy in producing optimal local alignments, these gapped extensions use by default a substantially larger Xg parameter than employed during the program’s search stage. » [3]

Le 27 octobre 2010
G.F. : Actually, if both Legacy Blast (based on C) and BLAST+ (partially based on C++) now share a common core-code branch in vanilla C, what is the difference between both softwares ?

T.M. : Good question. The main difference that occur to me right now are the BLAST database reader, the (report) formatting code, and the query splitting. The BLAST database reader may need to scan giga-bytes of data and deliver it to the blast engine for processing. We wanted to keep that separate from the core blast engine so it would be easier to replace, upgrade, etc. It really has nothing to do with the blast algorithm either and could be used with other apps. So, in the C toolkit we used the old database reader (readdb), but in the C++ toolkit we wrote a new one (CSeqDB). Partial fetching of subject sequences is only implemented in CSeqDB. The formatter in the C++ toolkit was rewritten from scratch, so it's different. Finally we implemented the query splitting in the C++ API directory. I think this decision was kind of questionable, but that's how it was in the end.

Annexe 3 : What's New Archive (1997)

http://www.ncbi.nlm.nih.gov/About/whatsnew97.html

09/10 PSI-BLAST A new service called Position-Specific Iterated BLAST, or PSI-BLAST, is now available in Version 2.0 of the BLAST program suite. Currently, PSI-BLAST may be used only for comparing protein queries with protein databases. Given a query, PSI-BLAST performs an initial gapped BLAST search of the database. In subsequent iterations, it uses statistically significant alignments from the previous search to construct a position-specific score matrix for use, in place of the query and standard amino acid substitution matrix, in the next round of searching. PSI-BLAST is under development and may change substantially over time.

08/19 Gaps in BLAST Version 2.0 of BLAST allows the introduction of gaps (deletions and insertions) into alignments. With a gapped alignment tool, homologous domains do not have to be broken into several segments. Also, the scoring of gapped results tends to be more biologically meaningful than ungapped results.

The programs, blastn and blastp, offer fully gapped alignments. blastx and tblastn have "in-frame" gapped alignments and use sum statistics to link alignments from different frames. tblastx provides only ungapped alignments.

08/12 Zea mays Maps The UMC (University of Missouri-Columbia RFLP Laboratory) and BNL (Brookhaven National Laboratory) maps are now available in the Entrez Genomes division. These maps are linked to a sequence map generated using available Zea mays sequences deposited in GenBank. Further information may be obtained at the Maize Genome Database.

06/26 PubMed Announced by Vice President Gore, NIH Officials The availability of PubMed was officially announced at a Capitol Hill press conference by Vice President Gore, Senators Harkin, and Specter and the Directors of NIH, NLM, and NCBI.

01/25 Complete Genome, E. coli The complete genome sequence and annotation of Escherichia coli as prepared by Fred Blattner and colleagues is now available via Entrez Genome division, GenBank, and the "nr" BLAST databases.

Annexe 4 : référence de Netblast (extrait)

http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/netblast.html

NOTICE:
The introduction of blast+ and the incorporation of "-remote" flag in all blast programs within the package provides a more stable way to perform batch BLAST searches remotely on NCBI server.

This makes netblast program obsolete. The instruction below is for your reference only.
May 17, 2010

Program Option for Netblast (blastcl3)
Tao Tao, Ph.D.
User Service
NCBI, NLM, NIH

1. Introduction

NCBI BLAST web server provides a convenient and user friendly way for individuals to search their queries against different public sequence databases. Even though this server can take multiple quereis and perform batch searches, true large scale batch searches may not go through if the input queries are long and the search settings are less stringent. In addition, the available databases for the web interface is somewaht limited. BLAST client provides a way to circumvent those limitations.

The BLAST client, or blastcl3, bypasses the web browser and interacts directly with the NCBI BLAST server that powers the NCBI web BLAST service (www.ncbi.nlm.nih.gov/BLAST/). It performs the batch search with multiple sequences by taking one query sequence at a time from the input file, formulating the search according to the settings of the command line parameters, and sending the search through the internet connection to NCBI BLAST server for processing. The program receives the search results from the BLAST server, in the format set in the search command line, and saves it to a local file specified. The program loops through all the queries in the input file until all are searched.

This program has no graphic user interface (GUI) and must be executed from command line under a terminal window. Users control the program through command line options. Detailed list of command line options are in Section 4. For usages and situation examples, see Section 3.

Annexe 5 : discontiguous MEGABLAST (1/2)

Annexe ajoutée en Mai 2013. [Consulté le 21.5.2013]

4.2 Discontiguous MEGABLAST is better at finding nucleotide sequences similar, but not identical, to your nucleotide query.

The BLAST nucleotide algorithm finds similar sequences by breaking the query into short subsequences called words. The program identifies the exact matches to the query words first (word hits). BLAST program then extends these word hits in multiple steps to generate the final gapped alignments.

One of the important parameters governing the sensitivity of BLAST searches is the length of the initial words, or word size as it is called. The most important reason that blastn is more sensitive than MEGABLAST is that it uses a shorter default word size (11). Because of this, blastn is better than MEGABLAST at finding alignments to related nucleotide sequences from other organisms. The word size is adjustable in blastn and can be reduced from the default value to a minimum of 7 to increase search sensitivity.

A more sensitive search can be achieved by using the newly introduced discontiguous megablast page. This page uses an algorithm with the same name, which is similar to that reported by Ma et.al. Rather than requiring exact word matches as seeds for alignment extension, discontiguous megablast uses non-contiguous word within a longer window of template. In coding mode, the third base wobbling is taken into consideration by focusing on finding matches at the first and second codon positions while ignoring the mismatches in the third position. Searching in discontiguous MEGABLAST using the same word size is more sensitive and efficient than standard blastn using the same word size. For this reason, it is now the recommended tool for this type of search. Alternative non-coding patterns can also be specified if desired. Additional details on discontiguous are available at : www.ncbi.nlm.nih.gov/Web/Newsltr/FallWinter02/blastlab.html

Parameters unique for discontiguous megablast are:

> word size: retricted to two options, i.e., 11 or 12
> template: only three options are available, 16, 18, or 21
> template type: coding (0), non-coding (1), or both (2)
> [Ajouter window_size (40)]

It is important to point out that nucleotide-nucleotide searches are not the best method for finding homologous protein coding regions in other organisms. That task is better accomplished by performing searches at the protein level, by direct protein-protein BLAST searches or by translated BLAST searches. This is because of the codon degeneracy, the greater information available in amino acid sequence, and the more sophisticated algorithm and scoring matrix used in protein-protein BLAST.

http://blast.ncbi.nlm.nih.gov/Blast.cgi?

CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=ProgSelectionGuide#blastn

Annexe 6 : discontiguous MEGABLAST (2/2)

Annexe ajoutée en Mai 2013.

What is discontiguous Mega BLAST ?

This version of Mega BLAST is designed specifically for comparison of diverged sequences, especially sequences from different organisms, which have alignments with low degree of identity, where the original Mega BLAST is not very effective. The major difference is in the use of the 'discontiguous word' approach to finding initial offset pairs, from which the gapped extension is then performed.

Both Mega BLAST and all previous versions of nucleotide-nucleotide BLAST look for exact matches of certain length as the starting points for gapped alignments. When comparing less conserved sequences, i.e. when the expected share of identity between them is e.g. 80% and below, this traditional approach becomes much less productive than for the higher degree of conservation. Depending on the length of the exact match to start the alignments from, it either misses a lot of statistically significant alignments, or on the contrary finds too many short random alignments.

According to [1], as well as our own probability simulations, it turns out that if initial 'words' are based not on the exact match, but on a match of a certainset of nonconsecutive positions within longer segments of the sequences, the productivity of the word finding algorithm is much higher. This way fewer words are found overall, but more of them end up producing statistically significant alignments, than in the case of contiguous words of the same, and even shorter length than the number of matched positions in the discontiguous word.

As an example, we can define a pattern (template) of 0s and 1s of length e.g. 21:

100101100101100101101. For each pair of offsets in the query and subject sequences that are being compared, we compare the 21 nucleotide segments in these sequences ending at these offsets, and require only those positions in those segments to match that correspond to the 1s in the above template.

There are several advantages in using this approach. First, the conditional probabilities of finding word hits satisfying discontiguous templates given the expected identity percentage in the alignments between two sequences, are higher than for contiguous words with the same number of positions required matched. If two word hits are required to initiate a gapped extension, the effect of the discontiguous word approach is even larger. In both cases higher

sensitivity is achieved because there is less correlation between successive words as the database sequence is scanned across the query sequence. Second, when comparing coding sequences, the conservation of the third nucleotides in every codon is not essential, so there is no need to require it when matching initial words. This implies the advantage of using templates based on the '110' pattern, which are called 'coding'. Finally, to achieve even higher sensitivity, one might combine two different discontiguous word templates and require any one of them to match at a given position to qualify it for the initial word hit.

The following options specific to this approach are supported:

* Template length: 16, 18, 21.
* Word size (i.e. number of 1s in the template): 11, 12
* Template type: coding, non-coding.
* Require two words for extension: yes/no.

The 'coding' templates are based on the 110 pattern, although more 0s are required for most of them, so some of the patterns become 010 or 100. These are the most effective for comparison of coding regions.

The non-coding templates attempt to minimize the correlation between successive words, when the database sequence is shifted by 4 positions against the query sequence. This means more 1s are concentrated at the ends of the template (at least 3 on each side).

When the option to require two words for extension is chosen, two word hits matching the template must be found within a distance of 50 nucleotides of one another.

Below are the exact discontiguous word template patterns for different combinations

of word sizes and lengths:

W = 11, t = 16, coding: 1101101101101101
W = 11, t = 16, non-coding: 1110010110110111
W = 12, t = 16, coding: 1111101101101101
W = 12, t = 16, non-coding: 1110110110110111
W = 11, t = 18, coding: 101101100101101101
W = 11, t = 18, non-coding: 111010010110010111
W = 12, t = 18, coding: 101101101101101101
W = 12, t = 18, non-coding: 111010110010110111
W = 11, t = 21, coding: 100101100101100101101
W = 11, t = 21, non-coding: 111010010100010010111
W = 12, t = 21, coding: 100101101101100101101
W = 12, t = 21, non-coding: 111010010110010010111

[1] Ma, B., Tromp, J., Li, M., "PatternHunter: faster and more sensitive

homology search", Bioinformatics 2002 Mar;18(3):440-5

http://www.ncbi.nlm.nih.gov/blast/discontiguous.shtml

Consulté le 21.5.2013

Annexe 7 : How does blastall find the alignment ?

Annexe ajoutée en Mai 2013.

1.2 How does blastall find the alignment ?

BLAST finds the optimal alignment by using the 'word matching algorithm', in which BLAST does the search in several distinctive phases:

1. generating overlapping words from the input query;
2. scanning the database for word matches (hits); and
3. extending word hits to produce (local) alignments through three steps of extension

During the first phase, BLAST breaks the input query into short overlapping segments, or 'words', and stores them in a hash table. BLAST takes those query words and scans the target database for initial matches in the second phase. The nucleotide BLAST algorithm looks for any single exact word match (with the exception of discontiguous mode in megablast). The protein BLAST algorithm uses a scoring matrix and a scoring threshold (neighborhood score threshold) to identify matches. Protein BLAST algorithm also requires two word hits within a certain distance in order to proceed to the next extension phase. In the third phase, initial matches or word hits are used as seeds of alignment extension. This phase is divided into three separate steps. The first step is the un-gapped extension, in which BLAST extends the initial word hits in both directions to generate initial alignments without allowing the introduction of gaps. The second step is the gapped extension, in which BLAST attempts to extend the initial un-gapped alignments further by allowing gapping in the new extension step. BLAST only keeps track of the start and end positions of a given alignment without saving the alignment details. In the last step, BLAST performs a final gapped extension with trace-back to generate the actual alignments.

It is important to note that limits on the number of alignments (-b), number of descriptions (-v), and e-value cut-off (-e) are applied at each of the three extension steps. Very stringent settings may cause BLAST to miss some good hits since the initial un-gapped HSPs may fall below cutoff and not be carried over to the gapped extension step.

http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/blastall/blastall_node4.html

Annexe 8 : dernières annonces du NCBI, BLAST+ 2.2.28, arrêt de Blastcl3, la collection « nt » par défaut comme base de recherche nucléotidique

Annexe ajoutée en Mai 2013.

From: "Mcginnis, Scott (NIH/NLM/NCBI) [E]" <mcginnis@ncbi.nlm.nih.gov>

To: NLM/NCBI List blast-announce <blast-announce@ncbi.nlm.nih.gov>

Thread-Topic: **BLAST+ 2.2.28 release**

Date: Mon, 1 Apr 2013 13:05:32 +0000

March 19, 2013

* 2.2.28 release.

New features:

> * Composition based statistics support in rpsblast
> * Support for taxonomy data in custom tabular output format
> * Support for query coverage and title in custom tabular output format
> * blastdbcmd support for batch subsequence retrieval

Improvements:

> * Adaptive BATCH_SIZE
> * Perform incremental XML output

Bug fixes:
* Segmentation fault on out-of-memory
* Prevented extension of alignment into Ns
* Replace tabs with spaces in FASTA deflines
* blastdbcmd displaying internal sequence ID for databases built without -parse_seqid
* blastdbcmd not fetching sequence data for complete sequence ID and -target_only
* blastn missing a hit for small word sizes
* Crash in blastn when it fetches sequence data from Genbank
* DeltaBLAST returning no hits when used with -remote option and searching more than one query
* psiblast problem using -import_search_strategy
* makeblastdb problem with ASN.1 input
* blastx reporting of HSPs dependent on -max_target_seqs
* psiblast's display of number of queries in tabular output format
(...)

From: "Mcginnis, Scott (NIH/NLM/NCBI) [E]" <mcginnis@ncbi.nlm.nih.gov>

To: NLM/NCBI List blast-announce <blast-announce@ncbi.nlm.nih.gov>

Thread-Topic: **Blastcl3 shutdown notice**

Date: Mon, 6 May 2013 16:13:23 +0000

The Netblast client (blastcl3) that has provided batch search access to the NCBI Web BLAST service will be discontinued on July 1, 2013.

The BLAST+ applications replace and improve upon the functions provided by blastcl3. Blastcl3 users should switch to BLAST+ as soon as possible. Locally installed BLAST+ applications can perform remote searches using the NCBI Web service when the -remote option is included on the command line. The BLAST+ remote service has a number of advantages over the blastcl3 application. Blastcl3 requires a persistent connection during the entire search, can only submit one query at a time, and is unable to return the BLAST Request ID (RID) used in the search. The BLAST+ remote service can submit multiple queries (from FASTA input) at once, poll for the results using the BLAST RID, and also print the RID in the BLAST report. Using the BLAST RID, it is possible to reformat the search locally with the blast_formatter application, reformat the search at the NCBI web site, or use analysis tools such as the BLAST treeview or the taxonomy report.

The BLAST+ applications are available at *ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/* Information on installing and running the BLAST+ applications is available at *http://www.ncbi.nlm.nih.gov/books/NBK1762/*

From: "Mcginnis, Scott (NIH/NLM/NCBI) [E]" <mcginnis@ncbi.nlm.nih.gov>

To: NLM/NCBI List blast-announce <blast-announce@ncbi.nlm.nih.gov>

Thread-Topic: **nt becomes default BLAST db**

Date: Mon, 26 Nov 2012 15:12:52 +0000

Starting November 26, 2012, the nucleotide collection (nt) will be the default nucleotide search database. The nucleotide collection consists of GenBank+EMBL+DDBJ+PDB+RefSeq sequences, but excludes EST, STS, GSS, WGS, TSA, patent sequences as well as phase 0, 1, and 2 HTGS sequences.

The nt database recently joined a list of other fast, indexed database searches offered by the NCBI that include the human G+T (genome plus transcript) and mouse G+T databases, as well as the human and mouse reference genome databases. The indexed databases are available when using megaBLAST. Indexed megaBLAST, at the NCBI BLAST web page, can search queries of a couple thousand bases against the 43 billion base nt database in a few seconds.

Indexed searches at the NCBI use an in-memory index described by Morgulis et al. and the megaBLAST algorithm.

THE BLAST GALAXY
...or 20 years of impact on Biology

Keywords
Bio-informatics ; BLAST ; similarity ; heuristic ; evolution ; NCBI

Summary
BLAST is a set of tools dedicated to the search of nucleic or proteic sequences in public databanks, using similarity. BLAST is available as a local command-line application and as a web server at NCBI and at a lot of other databank and databases in the world. Those softwares use an heuristic which provide a strong improvement in execution time over the previous dynamic programming methods. Since its first date of publication in 1990, BLAST was continuously improved. In particular the capability to insert gaps in a similar sequences alignment has been added, thus reporting the indels that appended in the evolutionary history of both sequences. The web server was also improved, as well as the on-line documentation. In 2010, BLAST is used as high as 300 000 times a weekday, underlining the primary importance of BLAST to the Biology. Similarity search using BLAST has now a 20 years impact in assessing phylogeny or protein functionality.

Recommandations pour lancer une recherche de similarités en 5 minutes

1. identifier la tâche à effectuer (par exemple, trouver des séquences de nucléotides similaires à une autre)

2. pour optimiser le traitement, se rendre sur le guide de sélection du programme approprié à la tâche identifiée (site du NCBI, NLM, NIH)
http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/producttable.html (§ 3.)

3. se demander si la tâche ne nécessite pas plus de 5 minutes de conception pour parvenir efficacement à l'objectif, par exemple en observant la forme de la séquence considérée et en comprenant ses caractéristiques propres le cas échéant. L'examen visuel de la requête peut être nécessaire.

4. se rendre sur l'interface Web du NCBI pour lancer la recherche
http://blast.ncbi.nlm.nih.gov/Blast.cgi

5. le cas échéant, analyser avec précaution les séquences utilisées dans les itérations de PSI-BLAST

La galaxie BLAST
20 ans d'utilisation de la méthode BLAST

Adresse de contact
guillaume.fernandezATalumni.u-psud.fr

Mots-clefs
Bioinformatique ; BLAST ; recherche par similarité ; heuristique ; évolution ; utilisation ; NCBI

Résumé
BLAST est un ensemble de programmes de recherche de séquences nucléotidiques et protéiques par similarité, qui interroge les banques publiques de données de génomes. BLAST est disponible *via* une interface web ou bien localement en ligne de commande. Ces logiciels utilisent un algorithme heuristique qui permet un gain de temps appréciable par rapport aux méthodes exactes publiées avant 1990, année de publication de BLAST. Le programme a fait l'objet de constantes améliorations depuis. Il a notamment été adapté pour permettre l'insertion de trous dans les séquences alignées, modélisant ainsi insertions et délétions issues de l'histoire évolutive des séquences. L'interface web s'est également adaptée et la documentation a fait l'objet de régulières améliorations. BLAST est utilisé plus de 300 000 fois par jour, ce qui souligne l'importance pour la biologie d'un tel logiciel. La recherche des séquences par similarité a permis 20 ans d'approches comparatives utiles pour la phylogénie ou l'étude des fonctions assurées par les protéines.