Plasma, un nouvel algorithme progressif pour l'alignement multiple de séquences

Vincent Derrien, Jean-Michel Richer, Jin-Kao Hao

Plasma, un nouvel algorithme progressif pour l'alignement multiple de séquences

Vincent Derrien, Jean-Michel Richer et Jin-Kao Hao

LERIA - Université d'Angers, 2 Bd Lavoisier, 49035 Angers
{derrien,richer,hao}@info.univ-angers.fr

Résumé

L'alignement multiple de séquences est un problème NP-complet important en bioinformatique. Plusieurs algorithmes existent, basés sur différentes heuristiques. Nous présentons ici Plasma, un nouvel algorithme utilisant le principe de la programmation dynamique, appliqué à des blocs de séquences. Plasma a été testé sur l'ensemble des jeux d'essais de Balibase. Les premiers résultats montrent que Plasma obtient le meilleur alignement pour plusieurs de ces jeux d'essais, mais également que les temps de calcul sont très faibles.

Abstract

Multiple sequence alignment constitutes a standard and fundamental tool in bioinformatics. Given its theoretical complexity (NP-Hard), many heuristics algorithms have been developed. In this paper, we present Plasma, a new algorithm that introduces the notion of aligning blocs of sequences via dynamic programming. Evaluations on the benchmarks of the well konwn Balibase database show that Plasma is able to find the best results for several instances.

1 Introduction

L'alignement de séquences d'ADN ou de protéines est une opération fondamentale en bioinformatique. L'alignement de plusieurs séquences consiste à mettre en regard les parties communes en insérant des brèches. Le problème d'alignement de séquences consiste à déterminer un alignement qui optimise un critère d'évaluation. L'alignement de k séquences est appelé alignement par paire pour $k = 2$, et alignement multiple pour $k > 2$.

L'alignement multiple de séquences est un préalable à la reconstruction phylogénétique, et il permet également l'identification des domaines communs ou des zones conservées, afin de découvrir des motifs et d'aider à la prédiction de fonctions de protéines non connues.

Un algorithme basé sur la programmation dynamique [10] permet d'aligner simplement deux séquences de manière optimale selon une fonction de score. Sa complexité pour des séquences de longueurs m et n est en $O(m.n)$. Cet algorithme peut être généralisé pour $k > 2$ séquences, mais dans la pratique sa complexité spaciale ne lui permet pas d'être utilisée pour des valeurs de k supérieures à 4. Le problème d'alignement multiple de séquences a été démontré NP-Complet [19]. Il existe de nombreuses méthodes permettant d'apporter des solutions au problème d'alignement multiple. Ces méthodes peuvent être classées suivant deux grandes approches : les méthodes d'alignement progressif et les méthodes d'alignement itératif.

Les méthodes progressives telles que Clustal W [16] n'alignent pas toutes les séquences simultanément. Celle-ci sont alignées progressivement par sous-groupes, suivant un ordre qui dépend de la similarité des unes par rapport aux autres. Les méthodes itératives comme SAGA [12] vont au contraire réaliser un alignement de toutes les séquences simultanément. Les algorithmes existant sont très différents les uns des autres, et sont plus lents que les algorithmes progressifs.

L'algorithme Plasma que nous présentons dans cet article est de type progressif, basé sur le principe de la programmation dynamique. D'autres algorithmes utilisent ce principe, mais ils sont basés sur la méthode utilisée pour l'alignement de deux séquences. Dès que l'algorithme nécessite d'aligner un groupe de séquences, avec une ou plusieurs autres séquences, ce groupe est transformé en un seule séquence consensus, appelée *profil*. Cette méthode est simple et rapide à utiliser, mais lorsque les séquences ne sont pas très si-

milaires, le profil est moins représentatif. Il provoque une perte d'information, et l'alignement qui en résulte risque donc d'être de mauvaise qualité.

Dans Plasma, nous cherchons à éviter ce problème en introduisant la notion d'alignement par blocs de séquences sans passer par les profils. La totalité de l'information contenue dans chacune des séquences est ainsi conservée. La méthode employée pour aligner deux blocs de séquences est une extension de l'algorithme de programmation dynamique telle qu'elle est utilisée pour l'alignement de deux séquences. En effet, Plasma permet d'aligner simultanément deux blocs de séquences qui ont déjà été préalablement alignées. L'algorithme permet de déterminer la meilleure insertion possible de colonnes de brèches dans chacun des deux blocs de séquences. En réitérant le processus, il est possible d'ajouter progressivement toutes les séquences, jusqu'à obtenir l'alignement complet.

Nous avons testé notre algorithme en utilisant Balibase, une base de données contenant près de 150 jeux d'essais classés par catégories. Pour chacun d'entre eux un alignement de référence est proposé, ce qui permet de tester les algorithmes mais aussi de les comparer entre eux.

Les premiers résultats obtenus avec notre implémentation sont encourageants puisque nous obtenons le meilleur résultat sur 7 jeux d'essais.

Dans la suite de cet article, nous exposons en détail le problème de l'alignement de séquences ainsi que quelques uns des algorithmes existants. Nous présentons ensuite Plasma, notre algorithme basé sur la programmation dynamique ainsi que les résultats obtenus sur un ensemble de jeux d'essais.

2 Alignement de séquences

2.1 Alignement par paire

L'alignement par paire de séquences de protéines est un outil fondamental de la bioinformatique. Il a pour but principal de faire ressortir les séquences apparentées, en mettant en évidence les régions communes. L'alignement par paire est principalement utilisé pour la comparaison d'une séquence avec un ensemble de séquences. Les algorithmes Fasta [13] et Blast [1] permettent de comparer une séquence à un ensemble de séquences contenues dans une base de données.

Une séquence peut être définie comme une série ordonnée de lettres prises dans un alphabet Σ. Pour les protéines, cet alphabet est constitué de 20 lettres, appelées acides aminés ou résidus.

Le concept d'alignement de séquences est basé sur la notion d'opérations d'édition [2]. Les opérations *match* ou *mismatch* correspondent à une mise en regard de deux caractères. Dans le premier cas il s'agit de résidus identiques, et dans le second de résidus différents. Les opérations *insertion* et *deletion* représentent une coupure, appelée brèche ou gap, dans l'une ou l'autre des séquences. Celle-ci est marquée par le symbole '$-$' dans l'alignement résultant.

Ces opérations permettent de définir un nombre exponentiel d'alignements, le problème consiste à déterminer le meilleur, au sens d'une fonction de score. Un coût est attribué à chaque opération d'édition au moyen d'une matrice de score w. Le coût d'un alignement est donné par la somme des coûts de chacune des opérations d'édition associées à cet alignement. Le problème d'alignement de 2 séquences peut donc être considéré comme un problème d'optimisation.

Définiton 1 *Soient* $S_1 = \langle x_{11}, x_{12}, \ldots, x_{1|S_1|}\rangle$ *et* $S_2 = \langle x_{21}, x_{22}, \ldots, x_{2|S_2|}\rangle$ *2 séquences définies sur un alphabet* Σ. *Un alignement* A *de* S_1 *et* S_2 *est une matrice de caractères de* $\Sigma \cup \{-\}$ *définie par :*

$$A = \begin{bmatrix} a_{11}, a_{12}, \ldots, a_{1q} \\ a_{21}, a_{22}, \ldots, a_{2q} \end{bmatrix}$$

et vérifiant les propriétés :
- $max(|S_1|, |S_2|) \leq q \leq |S_1| + |S_2|$,
- $a_{ui} = x_{uv}$ *ou* $-$, $\forall u \in \{1, 2\}, \forall v \in [1..|S_u|]$,
- $\nexists i$ *tel que* $a_{1i} = a_{2i} = -$.

Définiton 2 *La fonction de score d'un alignement par paire des deux séquences* S_1 *et* S_2 *est définie par :*

$$f(A) = \sum_{i=1}^{i=q} w(a_{1i}, a_{2i})$$

où w *désigne la matrice de score utilisée.*

Définiton 3 *Le problème d'alignement par paire consiste à construire un alignement qui optimise la fonction de score.*

Le problème d'alignement par paire peut être résolu en utilisant un algorithme de programmation dynamique ; celui ci permet d'obtenir l'alignement optimal en créant une matrice de scores. La complexité de cet algorithme est en $O(m.n)$, où m et n sont les longueurs des séquences à aligner. Le processus d'alignement de deux séquences peut être défini formellement de la façon suivante :

Définiton 4 *Soient* S_1 *et* S_2 *deux séquences de longueurs respectives* m *et* n *définies sur un alphabet* Σ. *Soient* $P(i, j)$ *le problème consistant à aligner les deux sous-séquences* $S_1[1..i]$ *et* $S_2[1..j]$, $i \leq m$ *et* $j \leq n$, *et* $D(i, j)$ *la distance d'édition associée.*
- *le problème consiste à déterminer* $P(m, n)$,

- *le sous-problème $P(i,j)$ consiste à calculer $D(i,j)$.*
- *l'initialisation se fait avec $P(i,0)$ et $P(0,j)$.*

Les problèmes $P(i,0)$ et $P(0,j)$ représentent le décalage d'une des séquences par rapport à l'autre, et leurs coûts sont simples à calculer.

2.2 Alignement multiple

Le problème d'alignement multiple correspond à une généralisation de l'alignement par paire avec $k > 2$ séquences. En revanche il ne s'agit plus ici de détecter une simple similitude entre séquences. L'alignement multiple de séquences est utilisé pour différentes opérations. Il permet de déterminer des sous-groupes de séquences en fonction du degré de similarité. Ce qui constitue le point de départ pour la reconstruction de phylogénie.

L'alignement multiple permet également de mettre en évidence les zones conservées dans un ensemble de séquences. En partant du principe que des motifs similaires induisent des fonctions identiques, l'alignement multiple permet de prédire la fonction de protéines inconnues en les alignant avec des protéines connues.

Définiton 5 *Soit $S = \{S_1, \ldots, S_k\}$ un ensemble de séquences définies sur un alphabet Σ. Un alignement multiple de S est une matrice d'éléments de $\Sigma \cup \{-\}$ définie par :*

$$A = \begin{bmatrix} a_{11}, a_{12}, \ldots, a_{1q} \\ \ddots \\ a_{k1}, a_{k2}, \ldots, a_{kq} \end{bmatrix}$$

et vérifiant les propriétés :
- $\max_i(|S_i|) \leq q \leq \sum_i |S_i|$,
- $a_{ui} = x_{uv}$ *ou* $-$, $\forall u \in \{1..k\}, \forall v \in [1..|S_u|]$,
- $\nexists j$ *tel que* $\forall i, a_{ij} = -$.

La fonction de score utilisée pour l'alignement par paire peut également être généralisée :

Définiton 6 *La fonction de somme des paires est définie par :*

$$f(A) = \sum_{i=1}^{i=k-1} \sum_{j=i+1}^{j=k} \sum_{l=1}^{l=q} w(a_{il}, a_{jl})$$

où w désigne la matrice de score utilisée.

Cette fonction est souvent utilisée sous une forme différente appelée somme des paires pondérée. Un poids p_i est associé à chacune des séquences en fonction de son importance dans l'alignement. Ce poids est ensuite pris en compte pour l'évaluation :

Définiton 7 *La fonction de somme des paires pondérée est définie par :*

$$f(A) = \sum_{i=1}^{i=k-1} \sum_{j=i+1}^{j=k} \sum_{l=1}^{l=q} p_i.p_j.w(a_{il}, a_{jl})$$

D'autres fonctions d'évaluation ont été développées depuis quelques années [9, 11, 18]. Les résultats qu'elles permettent d'obtenir dans certains cas semblent de meilleure qualité que ceux de la fonction de somme des paires pondérée.

Définiton 8 *Le problème d'alignement multiple de séquences consiste à construire un alignement qui optimise une de ces fonctions de score. Dans le cas de l'alignement multiple de séquences de protéines, il s'agit d'un problème de maximisation.*

La méthode de programmation dynamique peut également être utilisée pour réoudre le problème d'alignement multiple de séquences. Mais sa complexité spaciale est en $O(2^{k-1}.\prod_{i=1}^{i=k}|S_i|)$, ce qui en pratique ne permet pas de l'utiliser pour plus de 4 ou 5 séquences.

Le problème d'alignement multiple de séquences a été démontré NP-Complet [19] par réduction polynomiale du problème 1-to-3 SAT.

3 Etat de l'art

Différentes heuristiques ont été développées pour apporter des solutions au problème d'alignement multiple de séquences. Elles peuvent être classées suivant deux approches différentes. Une des approches est dite progressive, et consiste à aligner les séquences graduellement. Alors que la seconde, dite iterative, consiste à aligner toutes les séquences simultanément. Signalons également l'existence de méthodes complètes, qui permettent d'aligner quelques séquences.

Les résultats obtenus par ces différents algorithmes peuvent être comparés en utilisant les jeux d'essais proposés par des bases de données. Celles-ci sont constituées de nombreux jeux de séquences et proposent pour chacun un alignement de référence.

3.1 Les algorithmes progressifs

Les algorithmes progressifs construisent l'alignement final en plusieurs étapes. A chaque étape, une partie seulement des séquences est alignée, et ce n'est qu'à la fin que toutes les séquences se trouvent regroupées.

La méthode utilisée classiquement pour déterminer l'ordre dans lequel doivent être alignées les séquences est basée sur le principe du Neighbor-Joining (NJ) [15].

Pour cela, il faut réaliser les alignements par paire de tous les couples de séquences, afin de connaître leur degré de similarité. Il est ainsi possible de réaliser une matrice de "distances" entre toutes les séquences. Le Neighbor-Joining permet de créer un arbre, appelé *Guide Tree*, qui détermine l'ordre dans lequel s'effectue l'alignement. Le chemin remontant des branches vers la racine indique quels sont les groupes de séquences à aligner, ainsi que l'ordre dans lequel doivent se faire les alignements.

Le plus connu des algorihmes progressifs est *Clustal W* [16]. Son principe est basé sur l'algorithme de programmation dynamique appliqué à l'alignement de deux séquences [5]. Chaque alignement une fois obtenu est converti en une unique séquence consensus, appelée *profil*. La création d'un profil se fait en fonction du contenu de chacune des colonnes de l'alignement. Pour convertir l'alignement en une séquence, chaque colonne qui le compose est remplacée par une unique lettre. Le choix se fait en fonction du nombre d'occurrences de chaque lettre dans la colonne, ainsi que leur probabilité d'apparition. Un profil ainsi obtenu est considéré comme une séquence à part entière, et peut dès lors être réutilisé pour un nouvel alignement avec le même algorithme. Il peut être aligné avec une des séquences initiales, mais également avec un autre profil. Tous les noeuds internes constituant le *guide tree* représentent des profils.

Il existe d'autres algorithmes proposant des variantes de ce type d'alignement, comme par exemple T-Coffee [11] qui dans un premier temps commence par générer une bibliothèque d'alignements. A partir de cette bibliothèque, chaque couple de résidus se voit attribuer une valeur en fonction du nombre de fois où ils ont été alignés. Cette méthode permet d'éviter l'utilisation des matrices de coûts, dont les valeurs prévues pour le cas général, ne sont pas toujours adaptées.

Citons également Multalin [3] qui construit un nouveau *guide tree* à partir de l'alignement obtenu. Le processus d'alignement est réitéré tant que l'arbre obtenu est modifié.

3.2 Les algorithmes iteratifs

Les algorithmes de type itératif réalisent un alignement simultané de toutes les séquences. Ils n'utilisent pas de *guide tree* comme les algorithmes progressifs, ce qui les rend moins proches les uns des autres. Différentes approches ont ainsi été proposées, comme par exemple SAGA [12] qui est basé sur un algorithme génétique. Une popluation G_0 de cent alignements est initialement générée, et le programme permet de faire évoluer cette population d'alignements en les croisant entre eux et en ne conservant que ceux qui améliorent la fonction de score. Le schéma général de l'algorithme est le suivant :

- sélection de la partie de la population devant être remplacée,
- utilisation de l'un de nombreux opérateurs de combinaison ou de mutation sur chaque individu sélectionné,
- mise à jour de la population avec les meilleurs individus.

Ces trois opérations permettent de passer de la génération G_n à la génération G_{n+1}. L'algorithme s'arrête lorsque la population se stabilise. SAGA est l'algorithme itératif donnant les meilleurs résultats, quel que soit le type d'alignement. Le temps de calcul est toutefois beaucoup plus important que celui de Clustal W.

Citons également des approches basées sur les modèles de Markov cachés (HMM) [4], l'utilisation de graphe avec contraintes [8] ou encore un algorithme basé sur la méthode tabou [14].

Pour les alignements multiples composés de séquences assez similaires, les résultats obtenus par les algorithmes progressifs sont souvent meilleurs. Ils offrent également l'avantage d'être plus rapides.

3.3 Balibase

Afin d'évaluer les performances des différents algorithmes d'alignement, quelques bases de données ont été créées. Celles-ci proposent des jeux d'essais composés de séquences, ainsi qu'un alignement de référence.

Balibase [17] est une base de données contenant près de 150 alignements multiples de protéines. Ces alignements sont regroupés en cinq grandes catégories appelées références, chacune d'elles correspondant à une classe différente de problèmes. La référence 1 correspond à des jeux d'essais contenant peu de séquences. Elles sont classées par longueur (small, medium, long), et par pourcentage de similitude (< 20, < 40 et > 40). Les jeux d'essais de cette référence sont assez simples, et aucun algorithme de donne de très mauvais résultats. En revanche, les autres références sont composées de jeux d'essais plus atypiques. Citons par exemple, la présence d'une séquence orpheline, qui n'a aucune similarité avec les autres séquences ; mais également des séquences de tailles très différentes, ou nécessitant de très grandes brèches (longueur supérieure à 100).

Pour 139 de ces jeux d'essais, un meilleur alignement a été construit et est disponible. Ces alignements de référence ont été obtenus par les biologistes en prenant en compte des informations telles que la structure des séquences. Ils peuvent donc être considérés comme la solution optimale pour chacun des jeux d'essais, et les différents algorithmes peuvent ainsi être comparés entre eux. Nous présentons la méthode de comparaison des alignements dans la partie résultats de cet article.

4 Plasma

Dans Clustal W les séquences alignées sont remplacées par un profil. L'utilisation de ce profil fait disparaitre les séquences réelles, risquant ainsi une perte de la qualité. Avec Plasma nous proposons un nouvel algorithme de type progressif basé sur le principe de la programmation dynamique. Nous introduisons le concept de bloc qui nous permet de définir une méthode d'alignement utilisant toutes l'information des séquences.

4.1 Le principe de Plasma

L'utilisation d'un profil pour réaliser l'alignement multiple de séquences a l'avantage d'être une méthode simple et rapide à utiliser. Les résultats obtenus sont bons lorsque les séquences sont assez similaires. Toutefois le compromis qui doit être fait dans le cas d'une faible similarité engendre le risque de produire des profils de mauvaise qualité. Si le profil n'est pas représentatif des séquences à partir desquelles il est formé, l'alignement qui en découle risque de perdre en qualité. Le problème se répercutant au fur et à mesure des alignements.

L'algorithme de Plasma a été réalisé en partant de ce constat. Comme l'alignement risque de perdre en qualité à cause du profil, nous avons décidé de ne pas utiliser cette méthode. Les alignements sont effectués en conservant à chaque itération toutes les séquences.

Comme pour les autres algorithmes progressifs, Plasma commence par réaliser un *guide tree* en utilisant l'algorithme du Neigbour-Joining.

Le premier niveau d'alignement dans l'arbre correspond aux nœuds dont les fils représentent les séquences initiales. La méthode utilisée pour créer ces noeuds est celle de l'alignement par paire. Les résultats obtenus par cet algorithme ne sont en revanche pas convertis en profil. Nous conservons chaque alignement sous forme d'un "bloc". Le concept de bloc est défini à partir de l'alignement d'un sous-ensemble de séquences. Le bloc constitue une structure pour l'alignement, de sorte qu'il puisse être aligné par la suite. Lorsqu'un bloc est aligné, toutes les insertions de brèches se font entre les colonnes du blocs. Un bloc est un alignement qui peut à son tour être aligné avec d'autres séquences, mais dont le contenu de chaque colonnes reste inchangé. L'algorithme de Plasma va déterminer comment insérer des brèches à l'intérieur d'un bloc pour que l'alignement soit optimal. Ces brèches sont toujours insérées au début, à la fin ou entre deux colonnes du bloc.

La raison principale de l'utilisation de cette méthode vient de l'ordre d'alignement des séquences. Comme celui-ci se fait suivant une décroissance de la similarité entre les blocs, chaque paire de résidus préalablement alignés doit être conservée. Nous supposons donc ici que toute colonne précédemment obtenue doit être conservée. Ce raisonnement ne prend évidemment pas en compte les similarités qui peuvent exister localement entre des séquences distantes.

4.2 Utilisation de la programmation dynamique dans Plasma

L'algorithme de Plasma utilise la programmation dynamique pour réaliser l'alignement entre deux blocs ou entre un bloc et une séquence. Tous les alignements par paire restent effectués en utilisant l'algorithme traditionnel de programmation dynamique.

Le principe général de l'algorithme reprend celui existant pour deux séquences, mais étendu à l'alignement d'un bloc avec une séquence ou d'un bloc avec un autre bloc. La construction de l'alignement est réalisée en généralisant la méthode utilisée pour deux séquences et décrites dans [6].

Afin de réaliser l'alignement de deux blocs ou d'un bloc avec une séquence, nous avons redéfini les opérations d'édition. Pour l'alignement de deux séquences, ces opérations définissent la valeur associée aux différents couples de résidus possibles. La généralisation des opérations d'édition aux blocs nécessite de définir une valeur pour des couples de colonnes de résidus. La somme des valeurs obtenues pour chacune des opérations d'édition associées à l'alignement des deux blocs doit être égale à la valeur de l'alignement multiple.

Soit B un bloc de longueur l composé de n séquences définies sur un alphabet Σ. Nous noterons B_i la colonne i de ce bloc, et B_{ij} le contenu de la ligne j de B_i. Soit $\Sigma' = \Sigma \cup \{-\}$ et w la fonction définie sur $\Sigma' \times \Sigma'$ qui a deux éléments de Σ' associe leur évaluation.

Soit $B(p)$ l'évaluation de la colonne B_p.

$$B(p) = \sum_{i=1}^{i=n-1} \sum_{j=i+1}^{j=n} w(B_{pi}, B_{pj})$$

Les opérations *match* et *mismatch* étendues aux blocs, consistent à mettre en regard une colonne de chaque bloc. Soient B et B' deux blocs de longueur l et l' composées de n et n' séquences. La valeur associée à l'opération d'édition entre B_p et B'_q est donnée par :

$$M(B_p, B'_q) = B(p) + B'(q) + \sum_{i=1}^{i=n} \sum_{j=1}^{j=n'} w(B_{pi}, B'_{qj})$$

Les opérations *insertion* et *deletion* étendues aux blocs consistent à mettre en regard une colonne d'un

bloc avec une brèche. Dans le cas de l'insertion, la brèche est dans le premier bloc, la valeur associée est

$$I(B'_q) = B'(q) + n.\sum_{j=1}^{j=n'} w(-, B'_{qj})$$

De même pour la deletion :

$$D(B_p) = B(p) + n'.\sum_{i=1}^{i=n} w(B_{pi}, -)$$

Définiton 9 *Soient B_1 et B_2 deux blocs de séquences sur un alphabet Σ de longueurs respectives m et n. Soient $P(i,j)$ le problème consistant à aligner les deux sous-blocs $B_1[1..i]$ et $B_2[1..j]$, $i \leq m$ et $j \leq n$, et $D(i,j)$ la distance d'édition associée.*

- *le problème consiste à déterminer $P(m,n)$,*
- *le sous-problème $P(i,j)$ consiste à calculer $D(i,j)$.*
- *l'initialisation se fait avec $P(i,0)$ et $P(0,j)$.*

La construction récursive de la matrice D assure qu'à chaque étape, selon la fonction de sommes des paires pondérée, la valeur obtenue est optimale. En effet, les cas de base $P(i,0)$ et $P(0,j)$ correspondent à l'alignement d'un élément de Σ avec $-$. Ensuite le calcul du problème $P(i,j)$, est effectué à partir des sous-problèmes $P(i,j'), j' < j$ et $P(i',j), i' < i$. Or par hypothèse de récursivité, ces sous-problèmes ont une solution optimale.

Les brèches sont évaluées en utilisant la méthode affine classique. Le coût d'ouverture de la brèche est notée K et son coût d'extension h. Ainsi, une brèche de longueur l a une valeur $K+l.h$. Ce modèle d'évaluation impose d'utiliser la méthode décrite dans [6] pour aligner les blocs. Les calculs dans les différentes matrices sont réalisés avec les opérations d'édition étendues.

4.3 Algorithme général

L'algorithme général de Plasma se déroule en plusieurs étapes distinctes. Dans un premier temps Plasma calcul le *guide tree* associé aux séquences à aligner, ainsi que les pondérations associées à chacune d'elles.

Pour $n > 2$ séquences, l'algorithme va réaliser $n-1$ alignements. Il commence par réaliser les alignements par paire, correspondant dans le *guide tree* aux nœuds ayant pour fils deux séquences. Dans l'arbre, cela permet d'obtenir les nœuds de niveau 1. L'algorithme remonte ensuite progressivement jusqu'à la racine en construisant par alignement tous les nœuds intermédiaires. La dernière itération de l'algorithme regroupe toutes les séquences pour former l'alignement complet.

L'algorithme peut s'écrire :

```
procedure Aligner(E : liste de sequences)
debut
    Construire un guide tree de E par la méthode NJ
    Construire N blocs composés chacun d'une séquence
    L = {B_1, ..., B_n}
    Tant que |L| ≠ 1 faire
        Choisir les 2 blocs les plus proches B_i et B_j
        d'après le guide tree
        B = aligner(B_i, B_j)
        L = L ∪ {B_l}/{B_i, B_j}
    fin
```

4.4 La complexité

La complexité de l'algorithme est assez faible. Pour aligner deux blocs composés respectivement de n_1 et n_2 séquences et dont les longueurs sont l_1 et l_2, nous avons les complexités :

- $O(n_1^2.l_1$ et $n_2^2.l_2)$ pour les calculs à l'intérieur des blocs,
- $O(l_1.l_2.n_1.n_2)$ pour la construction des matrices,
- $O(l_1 + l_2)$ pour la construction de l'alignement.

Cette faible complexité permet à l'algorithme d'aligner très rapidement les séquences. Le temps de calcul est voisin de l'ordre de la seconde pour une vingtaine de séquences de longueur 200.

4.5 Implémentation de Plasma

L'implémentation a été réalisée en C++ et fonctionne sur une machine Linux. Elle est utilisable en ligne de commande, et permet de prendre en argument des paramètres standards généralement utilisés par les autres applications tels que la matrice de scores, mais également les coûts d'ouverture et d'extension de brèches K et h.

5 Résultats et discussion

5.1 Utilisation de Balibase

5.1.1 Les jeux d'essais

Pour évaluer notre algorithme nous avons utilisé les jeux d'essais proposés par Balibase (http ://www-igbmc.u-strasbg.fr/BioInfo/BAliBASE/). A cette adresse sont disponibles les jeux répartis en cinq références de problèmes, ainsi que la fonction de score permettant de comparer l'alignement d'un des jeux d'essai avec son alignement de référence.

5.1.2 Les critères d'évaluation

Cette fonction permet pour chaque jeu d'essai, de comparer le résultat d'un algorithme à l'alignement de

référence proposé par Balibase. Pour évaluer le résultat obtenu sur un jeu d'essai, cette fonction prend en paramètres l'alignement de référence ainsi que le résultat de l'algorithme. Deux évaluations sont retournées à l'utilisateur. La première représente le pourcentage de paires de résidus de l'alignement de référence contenus dans le résultat. La deuxième évaluation est le pourcentage de colonnes de l'alignement de référence contenues dans l'alignement résultat. Celle-ci est donc plus pénalisante puisqu'il suffit d'une erreur dans une colonne pour qu'elle soit fausse. En particulier dans la référence 2, la séquence orpheline est généralement assez mal alignée. Les valeurs obtenues sont souvent mauvaises alors que les autres séquences sont bien alignées. La première fonction d'évaluation permet de mettre en évidence la qualité globale de l'alignement, même si une seule séquence est mal alignée.

Le site Internet de Balibase propose également les résultats obtenus par 10 algorithmes progressifs et iteratifs. Les performances de tout nouvel algorithme peuvent ainsi être comparées à celles obtenues par des algorithmes connus. Ces résultats sont ceux obtenus avec l'évaluation du pourcentage de colonnes identiques.

Clustal W est aujourd'hui l'algorithme qui donne les meilleurs résultats sur l'ensemble des jeux d'essais de Balibase. Toutefois c'est sur les alignements avec séquences similaires qu'il est le plus performant. Les résultats obtenus par SAGA sont également bons sur l'ensemble des jeux d'essais. Les autres algorithmes semblent quant à eux plus destinés à une catégorie de problèmes.

5.2 Les paramètres de Plasma

Plasma nécessite quelques paramètres dont le premier est la matrice de score utilisée pour évaluer chaque paires de résidus. C'est principalement de cette matrice dont va dépendre le score. L'usage de certaines matrices standards, telles que PAM ou Blosum [7], est préconisé suivant le degré de similitude des séquences à aligner. Les deux autres paramètres importants de l'algorithme concernent l'évaluation des brèches. Pour éviter un nombre trop important de brèches dans les alignements, celles-ci doivent être pénalisées. Un coût négatif est donc associé aux brèches, en fonction de leur longueur. Le modèle linéaire est simple à utiliser, mais ne donne pas de résultats satisfaisants. De nombreux algorithmes utilisent un modèle affine, avec un coût K pour le début de la brèche (coût d'ouverture) et un coût h pour l'extension. Dans Plasma nous utilisons l'évaluation affine des brèches : $K+l.h$ où l représente la longueur de la brèche. Comme pour les matrices de score des paramètres K et h standards existent. Les valeurs à utiliser pour ces paramètres doivent également être précisés à Plasma.

Pour réaliser les tests sur les jeux d'essais de Balibase, nous avons utilisé des paramètres standards aussi bien pour la matrice [7] de scores que pour l'évaluation des brèches en prenant des valeurs de K voisines de -10 et des valeurs de h voisines de -1.

5.3 Les résultats

Plasma obtient le meilleur alignement pour 7 des 139 jeux de Balibase qui sont évalués avec la deuxième fonction de score [Tab 1].

Pour la première méthode d'évaluation, nous n'avons que les moyennes globales obtenues sur l'ensemble des jeux d'essais [14]. Au vu des résultats [Tab 2], Plasma se trouve en troisième position pour la qualité des alignements, juste derrière SAGA et Clustal W.

A partir des valeurs disponibles sur le site de Balibase, nous avons également calculé les moyennes sur chaque référence de jeux d'essais . Ces résultats sont ceux obtenus en utilisant la deuxième fonction de comparaison. Les moyennes obtenues par Plasma sont assez satisfaisantes avec cette méthode de comparaison. Notons en particulier que sur les références 3 et 4 les moyennes des résultats sont meilleures que celles de Clustal W. Notons que les valeurs obtenues par la algorithmes Multal et HMMT ne sont pas disponibles pour tous les jeux d'essais.

Signalons enfin que les tests ont été réalisés sur un ordinateur P4 2.4GHz. Les temps de calcul sont voisins de ceux de Clustal W, très inférieurs à 1 secondes pour les jeux d'essais de la référence 1. Les jeux d'essais les plus importants contiennent plus de 20 séquences et nécessitent quelques secondes de calcul.

5.4 Exemple d'alignement

Nous présentons ici en exemple un des alignements [Fig. 1] pour lequel Plasma obtient le meilleur résultat. Les deux colonnes contenant des erreurs d'alignement sont marquées avec le symbole '!'. Pour obtenir l'alignement de référence, il suffit dans ces deux colonnes de décaler le `AA` de la première séquence et le `V` de la deuxième séquence à gauche de la brèche. Ce résultat a été obtenu avec la matrice *Blosum 62* et les paramètres $K = -10$ et $h = -1$.

Après Plasma, Clustal W est l'algorithme qui obtient le meilleur résultat. Nous l'avons utilisé sur ce jeu d'essai pour voir les différences. Le résultat obtenu avec Clustal W donne la même erreur que Plasma au niveau de l'alignement, et nous avons indiqué avec un '?' les deux autres erreurs. Il s'agit de brèches dont

	PRRP	Clustal W	SAGA	Dialign	SB_Pima	ML_Pima	Multalign	Pileup8	Multal	HMMT	Plasma
1ubi	0,498	0,415	0,452	0	0,37	0,493	0,488	0,428	0,428	0,140	**0,525**
1dox	0,887	0,919	0,879	0,859	0,868	0,868	0,799	0,812	0,48	0,806	**0,963**
2mhr	0,98	0,985	0,952	0,951	0,975	0,908	0,962	0,965	0,961	0,803	**0,991**
1ezm	0,941	0,948	0,932	0,911	0,956	0,956	0,958	0,948	0,610	0,851	**0,975**
1pysA	0,931	0,922	0,906	0,936	0,935	0,935	0,926	0,931	0,917	0,616	**0,97**
1uky	0,139	0,13	0,269	0,139	0,037	0,083	0,148	0,241	0,083		**0,327**
1ajsA	0,128	0,163	0,186	0	0,006	0	0	0	0,11		**0,203**

TAB. 1 – Jeux d'essais pour lesquels Plasma obtient le meilleur résultat.

	PRRP	Clustal W	SAGA	Dialign	SB_Pima	ML_Pima	Multalign	Pileup8	Multal	HMMT	Plasma
Première fonction de comparaison											
Moy.	0,592	0,838	0,841	0,631	0,550	0,562	0,511	0,583			**0,799**
Deuxième fonction de comparaison											
Réf. 1	0,877	0,864	0,841	0,788	0,821	0,810	0,834	0,832	0,673	0,487	**0,750**
Réf. 2	0,541	0,583	0,586	0,384	0,401	0,379	0,371	0,517	0,429		**0,391**
Réf. 3	0,532	0,446	0,506	0,314	0,175	0,267	0,372	0,303	0,323		**0,463**
Réf. 4	0,323	0,361	0,289	0,853	0,794	0,705	0,292	0,710			**0,442**
Réf. 5	0,700	0,705	0,642	0,836	0,508	0,572	0,627	0,639			**0,643**

TAB. 2 – Résultats des différents algorithmes sur tous les jeux d'essais de Balibase.

```
hemt_sipcu      GFPVPDPFIW DASFKTFYDD LDNQHKQLFQ AILTQGNV-G GATAGDNAYA
1hrb            GFPIPDPYVW DPSFRTFYSI IDDEHKTLFN GIFHLAID-D NADNLGELRR
mp2_nerdi       GFEIPEPYKW DESFQVFYEK LDEEHKQIFN AIFALCGG-N NAGNLKSLVD
hem1_phago      -FDIPEPYVW DESFRVFYDN LDDEHKGLFK GVFNCAADMS SAGNLKHLID
hemt_linun      --KVPEPFAW NESFATSYKN IDLEHRTLFN GLFALSEF-N TRDQLLACKE
                                                        ?

hemt_sipcu      CLVAHFLFEE AAMQV-AKYG GYGAHKAAHE EFLGKVKGGS AD-----AAY
1hrb            CTGKHFLNQE VLMEA-SQYF -YDEHKKEHD GFINALDNWK GD-----VKW
mp2_nerdi       VTANHFADEE AMLKASASYG DFDSHKKKHE DFLAVIRGLG APVPQDKINY
hem1_phago      VTTTHFRNEE AMMDA-AKYE NVVPHKQMHK DFLAKLGGLK APLDQGTIDY
hemt_linun      VFVMHFRDEQ GQMEK-ANYE HFEEHRGIHE GFLEKMGHWK APVAQKDIKF
                                    ?                           !!

hemt_sipcu      CKDWLTQHIK TIDFKYKGK
1hrb            AKAWLVNHIK TIDFK-KGK
mp2_nerdi       AKEWLVNHIK GTDFGYKGK
hem1_phago      AKDWLVQHIK TTDFKYKGK
hemt_linun      GMEWLVNHIP TEDFKYKGK
```

FIG. 1 – Résultat obtenu pour l'alignement du jeu 2mhr.

la position est mauvaise, et qui sont situées dans des zones où les séquences sont faiblement similaires.

5.5 Discussion

Clustal W intègre des spécificités propres aux acides aminés lors de la construction des alignements. Par exemple un coût plus important pour les brèches assez proches ou encore éviter de séparer certaines combinaisons d'acides aminés.

Bien que Plasma n'intègre pas ces paramètres, il est en mesure de fournir de meilleurs résultats sur certains jeux d'essais. En comparant nos résultats avec ceux de Balibase, Plasma est meilleur que Clustal W sur 23 jeux d'essais.

6 Conclusion et perspectives

Dans cet article nous avons présenté le problème de l'alignement multiple de séquences ainsi que quelqu'un des algorithmes existants. Les algorithmes progressifs utilisent un profil pour réaliser l'alignement multiple. L'imprécision du profil étant d'autant plus importante que les séquences à aligner sont distantes.

L'algorithme Plasma que nous avons présenté vise à palier aux problèmes engendrés par l'utilisation de profils. Il est fondé sur la notion centrale d'alignement par bloc et emploie une extension de la programmation dynamique traditionnellement utilisée pour l'alignement par paire.

Le test de l'algorithme sur les instances de Balibase permet d'obtenir le meilleur résultat sur quelques uns des jeux d'essais.

La programmation dynamique lui permet d'être rapide pour calculer les résultats, aussi envisageons nous d'ajouter quelques fonctionnalités.

Nous avons utilisé dans notre implémentation la fonction de somme des paires pondérées pour évaluer la qualité de l'alignement. Le résultat obtenu par Plasma est directement lié à l'évaluation donnée par cette fonction lors de l'alignement de deux blocs. La méthode utilisée dans T-Coffee [11] permet d'avoir une fonction de score adaptée à chaque jeu d'essai. Nous envisageons de l'utiliser afin d'augmenter la qualité de l'alignement.

Une autre piste intéressante consiste à modifier localement un bloc après alignement. La conservation des colonnes permet d'utiliser la programmation dynamique pour l'alignement des blocs. Dans la pratique, certaines brèches à l'intérieur des blocs ne sont pas placées de façon optimale. Nous envisageons donc une redistribution locale des brèches après obtention d'un bloc.

7 Remerciements

Ce travail s'inscrit dans le cadre de Ouest Génopole du projet CER Post-Génome. A ce titre, il a été financé en partie par le Ministère de la Recherche, la région des Pays de la Loire et les collectivités locales.

Références

[1] S.F. Altschul, W. Gish, W Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, Vol. 215(3) :403–410, 1990.

[2] P. Clote and R. Backofen. *Computational Molecular Biology - An Introduction*. Mathematical and Computational Biology. Wiley, 2000.

[3] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, Vol. 22 :10881–10890.

[4] S.R. Eddy. Multiple alignment using hidden markov models. In CA AAAI Press, Menlo Park, editor, *Third International Conference on Intelligent Systems for Molecular Biology (ISBM)*, pages 114–120, 1995.

[5] D.F. Feng and R.F Doolittle. Progressive alignment and phylogenetic tree construction of protein sequences. *Methods in Enzymology*, 183 :375–387, 1990.

[6] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, Vol. 162 :705–708, 1982.

[7] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. In *Proceedings of the National Academy of Science*, volume Vol. 89, pages 10915–10919, 1992.

[8] J. D. Kececioglu, H-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, and M. Vingron. A polyhedral approach to sequence alignment problems. *Discrete Appl. Math.*, 104(1-3) :143–186, 2000.

[9] B. Morgenstern. Dialign2 : improvment of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, Vol. 15 (3) :211–218, 1999.

[10] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3) :443–453, 1970.

[11] C. Notredame, D. Higgins, and J. Heringa. T-coffee : A novel method for multiple sequence alignments. *Journal of Molecular Biology*, Vol. 302 :205–217, 2000.

[12] C. Notredame and D.G. Higgins. Saga : Sequence alignment by genetic algorithm. *Nucleic Acid Research*, Vol. 24 :1515–1524, 1996.

[13] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. In *Proc Natl Acad Sci U S A*, volume 85(8), pages 2444–2448, 1988.

[14] Tariq Riaz, Yi Wang, and Kuo-Bin Li. Multiple sequence alignment using tabu search. In *CRPIT '29 : Proceedings of the second conference on Asia-Pacific bioinformatics*, pages 223–232. Australian Computer Society, Inc., 2004.

[15] N. Saitou and M.Nei. The neighbor-joining method : a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4 :406–425, 1987.

[16] J.D. Thompson, D.G. Higgins, and T.J. Gibson. Clustal w : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, Vol. 22 :4673–4680, 1994.

[17] J.D. Thompson, F. Plewniak, and O. Poch. Balibase : A benchmark alignments database for the evaluation of multiple sequence alignment programs. *Bioinformatics*, Vol. 15 :87–88, 1999.

[18] J.D. Thompson, F. Plewniak, R. Ripp, J-C. Thierry, and O. Poch. Towards a reliable objective function for multiple sequence alignments. *Journal of Molecular Biology*, Vol 314 :937–951, 2001.

[19] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4) :337–348, 1994.