New methods for biological sequence alignment

Marta L. Gîrdea

▶ **To cite this version:**

Marta L. Gîrdea. New methods for biological sequence alignment. Bioinformatics [q-bio.QM]. Université des Sciences et Technologie de Lille - Lille I, 2010. English. NNT : . tel-00833311

HAL Id: tel-00833311
https://theses.hal.science/tel-00833311

Submitted on 12 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse de Marta Girdéa, Lille 1, 2010

New methods for biological sequence alignment

THÈSE

présentée et soutenue publiquement le 10 décembre 2010

pour l'obtention du

Doctorat de l'Université de Lille 1 – Sciences et Technologies

(spécialité informatique)

par

Marta GÎRDEA

Composition du jury

Rapporteurs :	Éric RIVALS, DR CNRS	LIRMM, Université Montpellier 2
	Michal ZIV-UKELSON, Professeur	Université Ben Gurion du Negev, Israel
Examinateurs :	Emmanuel BARILLOT, DR INRA	Institut Curie, Paris
	Michael BRUDNO, Professeur	Université de Toronto, Canada
	Clarisse DHAENENS, Professeur	Université Lille 1
Advisors :	Gregory KUCHEROV, DR CNRS	LIFL-INRIA, Université Lille 1
	Laurent NOÉ, Maître de conférences	LIFL-INRIA, Université Lille 1

UNIVERSITÉ DE LILLE 1 – SCIENCES ET TECHNOLOGIES
ÉCOLE DOCTORALE SCIENCES POUR L'INGÉNIEUR
Laboratoire d'Informatique Fondamentale de Lille — UMR 8022
U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX
Tél. : +33 (0)3 28 77 85 41 – Télécopie : +33 (0)3 28 77 85 37 – email : direction@lifl.fr

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Acknowledgements

I would like to thank my advisor Gregory Kucherov, whose expertise was very important in this research work, who has always managed to be present and stay involved regardless of any geographic distance, and who has taught me many things, from science to written and verbal communication.

I address a special *thank you* to my second advisor Laurent Noé for having closely guided and advised me during these three years, for his constant support, patience, friendliness, and active involvement which have played a crucial role in the progress of this thesis.

Various ideas related to the work presented here, either already included or planned as future developments, have emerged from interesting discussions with Emmanuel Barillot, Valentina Boeva, Michael Brudno, Martin Figeac, Mathieu Giraud, Maude Pupin, Éric Rivals, Hélène Touzet, Jean-Stéphane Varré, as well as from questions and suggestions raised by the anonymous reviewers from conferences and journals where we submitted article manuscripts.

I would then like to thank all the present and former members of SEQUOIA with whom I had the pleasure to interact during my three years in this team, for having created a work environment that was not only scientifically rich, but also particularly warm and friendly.

Last but not least, I would like to thank my former professor Liviu Ciortuz for having opened my interest towards this research field, and all my former professors at the "Al. I. Cuza" University of Iaşi for their contributions to my education.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Acknowledgements

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Abstract

Biological sequence alignment is a fundamental technique in bioinformatics, and consists of identifying series of similar (conserved) characters that appear in the same order in both sequences, and eventually deducing a set of modifications (substitutions, insertions and deletions) involved in the transformation of one sequence into the other. This technique allows one to infer, based on sequence similarity, if two or more biological sequences are potentially homologous, i.e. if they share a common ancestor, thus enabling the understanding of sequence evolution.

This thesis addresses sequence comparison problems in two different contexts: homology detection and high throughput DNA sequencing. The goal of this work is to develop sensitive alignment methods that provide solutions to the following two problems: i) the detection of hidden protein homologies *by protein sequence comparison*, when the source of the divergence are frameshift mutations, and ii) mapping short SOLiD reads (sequences of overlapping di-nucleotides encoded as colors) to a reference genome. *In both cases, the same general idea is applied: to implicitly compare DNA sequences for detecting changes occurring at this level, while manipulating, in practice, other representations (protein sequences, sequences of di-nucleotide codes) that provide additional information and thus help to improve the similarity search.* The aim is to design and implement exact and heuristic alignment methods, along with scoring schemes, adapted to these scenarios.

Résumé

L'alignement de séquences biologiques est une technique fondamentale en bioinformatique, et consiste à identifier des séries de caractères similaires (conservés) qui apparaissent dans le même ordre dans les deux séquences, et à inférer un ensemble de modifications (substitutions, insertions et suppressions) impliquées dans la transformation d'une séquence en l'autre. Cette technique permet de déduire, sur la base de la similarité de séquence, si deux ou plusieurs séquences biologiques sont potentiellement homologues, donc si elles partagent un ancêtre commun, permettant ainsi de mieux comprendre l'évolution des séquences.

Cette thèse aborde les problèmes de comparaison de séquences dans deux cadres différents: la détection d'homologies et le séquençage à haut débit. L'objectif de ce travail est de développer des méthodes d'alignement qui peuvent apporter des solutions aux deux problèmes suivants: i) la détection d'homologies cachées entre des protéines *par comparaison de séquences protéiques*, lorsque la source de leur divergence sont les mutations qui changent le cadre de lecture, et ii) le *mapping* de *reads* SOLiD (séquences de di-nucléotides chevauchantes codés par des couleurs) sur un génome de référence. *Dans les deux cas, la même idée générale est appliquée: comparer implicitement les séquences d'ADN pour la détection de changements qui se produisent à ce niveau, en manipulant, en pratique, d'autres représentations (séquences de protéines, séquences de codes di-nucléotides) qui fournissent des informations supplémentaires et qui aident à améliorer la recherche de similarités.* Le but est de concevoir et d'appliquer des méthodes exactes et heuristiques d'alignement, ainsi que des systemes de scores, adaptés à ces scénarios.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Abstract

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Table of contents

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

List of Figures

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

List of Tables

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

De nouvelles méthodes pour l'alignement des séquences biologiques

Introduction

L'alignement de séquences biologiques est une technique fondamentale en bioinformatique, et consiste à identifier des séries de caractères similaires (conservés) qui apparaissent dans le même ordre dans les deux séquences, et à inférer un ensemble de modifications (substitutions, insertions et suppressions) impliquées dans la transformation d'une séquence en l'autre. Cette technique permet de déduire, sur la base de la similarité de séquence, si deux ou plusieurs séquences biologiques sont potentiellement homologues, donc si elles partagent un ancêtre commun, permettant ainsi de mieux comprendre l'évolution des séquences.

Du point de vue computationnel, la comparaison de séquences biologiques est fortement liée à l'édition de chaînes de caractères, car les séquences biologiques sont généralement représentées comme des chaînes de caractères sur un certain alphabet: l'alphabet des quatre *nucléotides* dans le cas des séquences d'ADN, ou l'alphabet des vingt *acides aminés* pour les protéines. Le degré de similarité des séquences alignées est quantifiée par le *score* associé à leur alignement, qui comprend les scores associés (généralement de façon indépendante) aux couples de caractères alignés, tenant compte de leur similitude. En général, les algorithmes utilisent une approche basée sur la *programmation dynamique* afin d'obtenir le meilleur alignement de séquences par rapport à un certain système de score. Les premiers algorithmes d'alignement [75, 150, 188] et les systèmes de score [51, 86] ont été proposés il y a plusieurs décennies, et sont toujours utilisés en pratique. Plus tard, l'émergence des technologies de séquençage [170, 176, 181] conduit à l'accumulation d'une quantité impressionnante de données qui ont besoin d'être analysées. Ainsi, en complément des algorithmes exacts d'alignement, des approches heuristiques plus rapides [11, 130, 164] ont été développées afin de faire face aux séquences de grande taille dont l'alignement exact exigerait un temps de calcul excessif. Ces heuristiques sont basées sur l'hypothèse que des séquences suffisamment liées doivent partager au moins une région bien conservée, appelé *graine*. L'approche consiste essentiellement à identifier cette région si elle existe, et ensuite à construire d'un alignement optimal autour d'elle.

Cette "recette" algorithmique générique convient aux problèmes classiques de comparaison de séquences d'ADN ou de protéines. Néanmoins, avec les quantités croissantes de données disponibles et les nouvelles applications, viennent des problèmes spécifiques qui peuvent nécessiter d'adapter de nombreux aspects de ce cadre général à leurs caractéristiques. De nombreuses approches développées par la suite ont montré que, par exemple, les procédures de traitement de paires de symboles alignés et de traitement des *gaps* au sein de l'algorithme

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

d'alignement [15,16,39,79,85,106,113,166,168,214], le système de scores [40,65,167], ou bien les similarités considérées comme "pertinentes" dans les alignements basées sur des graines peuvent être ajustés pour tenir compte des connaissances sur un certain problème et ainsi capturer le degré de similitude entre les structures comparées. En outre, une grande attention a été donnée à la *conception* des graines, afin de se conformer à des modèles qui donnent les caractéristiques des séquences alignées [30, 31, 99, 105, 110, 133].

Cette thèse aborde les problèmes de comparaison de séquences dans deux cadres différents: la détection d'homologies et le séquençage à haut débit.

La comparaison de séquences en tant qu'outil pour la détection d'homologies s'appuie sur le fait connu que la duplication de gènes suivie d'une divergence joue un rôle important dans l'évolution [4]. "*Natural selection merely modified, while redundancy created*" [158] est une hypothèse formulée il y a plusieurs décennies dans une tentative d'expliquer les progrès majeurs dans l'évolution, tels que la transition des formes de vie unicellulaires vers des organismes complexes [145]. Plus tard, des études sur les génomes de plusieurs organismes, entièrement ou presque entièrement séquencés, ont montré qu'une grande proportion de gènes ont en effet été générés par duplication, tandis que l'origine d'autres est soupçonnée résider dans ce mécanisme, mais la preuve est entravée par leur divergence avancée [213]. Ainsi, la similarité de séquence est considérée comme un indice fort de l'homologie, mais elle ne garantit pas l'homologie. Inversement, des séquences qui paraissent différentes ne sont pas nécessairement indépendantes. D'où *la nécessité de mettre en place des méthodes de comparaison assez puissantes pour révéler des similitudes de séquences qui sont réellement pertinentes pour détecter l'homologie.*

La divergence de séquences après duplication est le résultat de divers types de *mutations*, telles que les substitutions, insertions, ou délétions de nucléotides. Si elles affectent une séquence codante, c'est à dire un morceau d'ADN qui code pour une protéine ayant une certaine fonction dans l'organisme, ces modifications ont implicitement des effets sur le produit de cette séquence, la protéine traduite, et elles peuvent en modifier la fonction ou la rendre non fonctionnelle. Un problème de détection d'homologies particulièrement intéressant consiste à trouver des similitudes entre les protéines ayant divergé suite à un *frameshift*. Les *frameshifts* sont des insertions ou des suppressions de plusieurs nucléotides d'une séquence d'ADN codante. Une seule mutation *frameshift* introduit un changement radical dans la protéine traduite, tout en ayant un faible effet sur la séquence d'ADN touché, qui resté à première vue majoritairement inchangé. Toutefois, si des substitutions de nucléotides sont impliquées dans la divergence, la similitude au niveau de l'ADN peut être elle aussi réduite au-delà du seuil de reconnaissance. Suite aux changements importants survenant dans les séquences d'ADN et leurs traductions, les algorithmes d'alignement classiques sont alors susceptibles d'échouer à détecter l'homologie, aussi bien au niveau de l'ADN qu'au niveau de la séquence protéique. Des solutions algorithmiques pour résoudre ce problème sont proposées dans la partie **"Découverte de *frameshifts* dans les protéines en utilisant la traduction inverse"** de cette thèse.

La détection d'homologies n'est pas le seul contexte où l'analyse de séquences biologiques est basée sur la comparaison de séquences. Les méthodes d'alignement sont un ingrédient clé dans l'*assemblage* ou le *mapping sur un génome de référence* de "*reads*" produits par les technologies de séquençage à haut débit. Les *reads* sont des morceaux de la séquence cible, qui peuvent avoir de quelques dizaines à des centaines de symboles. Dans le contexte du *read mapping*, l'un des défis consiste à identifier la position candidate sur la séquence de référence pour chaque *read*. Cette dernière exige essentiellement de savoir détecter rapidement des similarités pertinentes entre le *read* et la référence. L'identification, ainsi que l'alignement lui-même du *read* sur la séquence de référence, doivent faire face non seulement aux variations biologiques, mais aussi

http://doc.univ-lille1.fr

aux erreurs de séquençage, ainsi qu'aux artefacts technologiques qui peuvent modifier le *read*.

La plate-forme de séquençage SOLiD [1] utilise un encodage particulier ayant la capacité de corriger certaines erreurs afin d'améliorer la distinction entre les erreurs de séquençage et les variations biologiques. Ainsi, au lieu d'être une séquence de symboles de nucléotides, un *read* SOLiD est une séquence de couleurs, chaque couleur codant pour deux nucléotides consécutifs et chaque nucléotide étant encodé par deux couleurs consécutives dans la séquence. En raison de double codage des nucléotides en couleurs adjacentes, les positions d'un *read* ne peuvent pas être interprétée de manière indépendante, ce qui empêche les algorithmes d'alignement classiques de capturer le similarités au niveau des séquences ADN par simple alignement de séquences de couleurs. Par contre, les variations biologiques et les erreurs de lecture peuvent être reconnues et distinguées en analysant simultanément plusieurs couleurs consécutives. Ce problème est abordé dans la partie "**Algorithmes pour le** *mapping* **de** *reads* **SOLiD**" de la thèse, qui traite des méthodes adaptées à ces particularités.

Pour résumer, l'objectif de ce travail est de développer des méthodes d'alignement qui peuvent apporter des solutions aux deux problèmes suivants: i) la détection d'homologies cachées entre des protéines *par comparaison de séquences protéiques*, lorsque la source de leur divergence sont les mutations qui changent le cadre de lecture, et ii) le *mapping* de *reads* SOLiD (séquences de di-nucléotides chevauchants codés par des couleurs) sur un génome de référence. *Dans les deux cas, la même idée générale est appliquée: implicitement comparer les séquences d'ADN pour la détection changements qui se produisent à ce niveau, lors de la manipulation, en pratique, d'autres représentations (séquences de protéines, des séquences de codes di-nucléotides) qui fournissent des informations supplémentaires et qui aident à améliorer la recherche de similarités.* Le but est de concevoir et appliquer des méthodes exactes et heuristiques d'alignement, ainsi que des systèmes de scores et des techniques d'évaluation, adaptés à ces scénarios.

Découverte de *frameshifts* dans les protéines en utilisant la traduction inverse

Dans le cadre de la *découverte d'homologies cachés entre des protéines lorsque la divergence est causée par un frameshift, par utilisation directe de la comparaison de séquences protéiques*, le scénario suivant de "duplication et divergence" est considéré. Un gène codant pour une protéine est dupliqué. Un des deux exemplaires subit une mutation *frameshift*. Par la suite, les deux copies peuvent être affectés par des substitutions de nucléotides, la plupart étant synonymes pour l'une des copies, afin de préserver la fonction, mais sans restriction pour la copie dont le cadre de lecture a changé.

Comme expliqué précédemment, sous cette hypothèse, l'origine commune des deux séquences peut être difficile à retracer après une longue période d'évolution, en raison des modifications à la fois sur les séquences d'ADN et sur les protéines encodées par ces séquences. Pour répondre à ce problème, nous proposons ici une méthode [68–70] basée sur des modèles d'évolution des codons, qui combine les connaissances sur les séquences d'ADN et de protéines afin de saisir la dynamique des séquences codantes. La méthode peut être utilisée pour plusieurs tâches de comparaison des protéines, mais convient notamment à deux situations particulières:

- Quand les protéines sont directement obtenues par spectrométrie de masse, et les séquences d'ADN correspondantes ne sont donc pas connues.
- Quand l'ADN codant est disponible, mais le degré de divergence est tel que la simple comparaison de séquences d'ADN ne peut pas montrer leur similarité.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

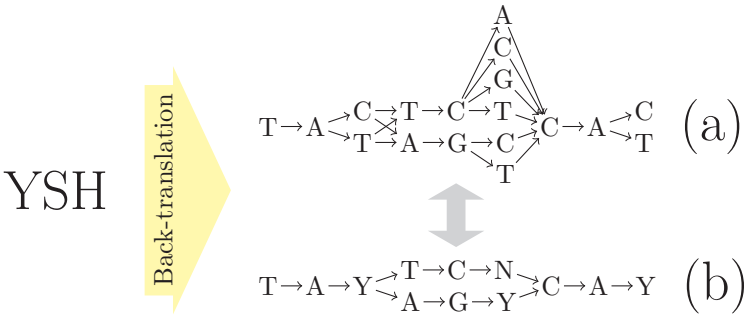

Figure 1: Exemple de graphe de traduction inverse pour la séquence d'acides aminés YSH, décrivant toutes les séquences d'ADN qui peuvent encoder ces acides aminés.

Cette approche permet également de détecter des homologies de protéines moins lointaines, avec ou sans *frameshifts*, ou bien des erreurs de séquençage de type insertion / suppression. Cependant, ces derniers cas d'utilisation ne sont pas son objectif principal.

Graphes de traduction inverse

La méthode proposée s'appuie sur la notion de *traduction inverse* de protéines, qui permet d'obtenir l'ensemble des séquences d'ADN codant pour une protéine donnée sous la forme compacte d'un *graphe de traduction inverse*. Ce dernier est un graphe acyclique dirigé, avec des nœuds étiquetés par des symboles de nucléotides, et dont les chemins décrivent les séquences d'ADN putatives. Comme l'illustre la Figure 1, le graphe est en fait organisé en $3n$ positions, où n est la longueur de la séquence protéique. Chaque position i du graphe a un ou plusieurs nœuds correspondants, et chaque nœud représente un nucléotide qui peut apparaître à une position i dans au moins l'une des séquences codantes putatives. Pour un graphe plus compact, les codes IUPAC d'ambiguïté de nucléotides permettent de représenter tous les acides aminés codés par 2-4 codons avec une seule séquence de 3 symboles, où les 3ème symbole peut comprendre 1, 2, 3 ou 4 nucléotides. Les acides aminés encodés par 6 codons peuvent être représentés par deux segments de 3 symboles.

Algorithme pour l'alignement de deux graphes de traduction inverse

Étant donnés deux graphes G_A et G_B obtenus par traduction inverse des protéines P_A et P_B, l'algorithme trouve le meilleur alignement local entre deux séquences d'ADN comprises dans ces graphes. Les solutions partielles sont stockées dans une table de programmation dynamique M. Pour chaque entrée $M[i, j, (\alpha_i, \beta_j)]$, i et j sont des positions respectives dans les graphes G_A et G_B. α_i et β_j énumèrent les paires possibles de nœuds qui se trouvent respectivement dans le graphe G_A à la position i, et dans le graphe G_B à la position j. En conséquence, $M[i, j, (\alpha_i, \beta_j)]$ contient le score du meilleur alignement entre les deux préfixes $D^A_{1..i}$ et $D^B_{1..j}$ des séquences d'ADN D^A et D^B avec $traduction(D^A) = P_A$ et $traduction(D^B) = P_B$, tel que $D^A_i \in \alpha_i$ et $D^B_j \in \beta_j$.

L'algorithme de programmation dynamique commence avec une **initialisation** de M qui est classique pour l'alignement local. L'étape de **récursion** est décrite par la relation (1). Les *frameshifts* sont traitées par l'algorithme à l'aide d'une fonction non-monotone de pénalité des *gaps*, où les insertions et suppressions de codons complets sont moins pénalisées que les changements de cadre de lecture. En outre, puisque les *frameshifts* sont considérés comme des événements très rares, une extension ce cet algorithme donne la possibilité de limiter leur nombre

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

dans un alignement.

$$
\begin{aligned}
&M[i,j,(\alpha_i,\beta_j)] = \\
&\max \begin{cases}
0 & & \text{(a)} \\
M[i-1,j-1,(\alpha_{i-1},\beta_{i-1})] + \sigma(\alpha_i,\beta_j), & \alpha_{i-1} \in pred_{G_A}(\alpha_i); \ \beta_{j-1} \in pred_{G_B}(\beta_j); & \text{(b)} \\
M[i,j-1,(\alpha_i,\beta_{j-1})] + fsGapPenalty, & \beta_{j-1} \in pred_{G_B}(\beta_j); & \text{(c)} \\
M[i,j-1,(\alpha_i,\beta_{j-1})] + fsExtensionPenalty, \ \textit{(seulement si l'entrée précédente sur le chemin de l'alignement est dans } M[i,j-2]\textit{)} & \beta_{j-1} \in pred_{G_B}(\beta_j); & \text{(c}') \\
M[i-1,j,(\alpha_{i-1},\beta_j)] + fsGapPenalty, & \alpha_{i-1} \in pred_{G_A}(\alpha_i); & \text{(d)} \\
M[i-1,j,(\alpha_{i-1},\beta_j)] + fsExtensionPenalty, \ \textit{(seulement si l'entrée précédente sur le chemin de l'alignement est dans } M[i-2,j]\textit{)} & \alpha_{i-1} \in pred_{G_A}(\alpha_i); & \text{(d}') \\
M[i,j-3,(\alpha_i,\beta_{j-3})] + tripleGapPenalty, & j \geq 3, j \text{ multiple de } 3; & \text{(e)} \\
M[i-3,j,(\alpha_{i-3},\beta_j)] + tripleGapPenalty, & i \geq 3, i \text{ multiple de } 3;. & \text{(f)}
\end{cases} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (1)
\end{aligned}
$$

Système de scores

L'algorithme utilise un *système de scores* conçu pour refléter le processus réel d'évolution, basé sur un modèle de substitution de codons. Les scores sont définis sur des paires de triplets (α, p, a), où α est un symbole de nucléotide, p est sa position dans le codon, et a est l'acide aminé encodé par ce codon. Ainsi, les différents contextes de la substitution sont clairement différenciés. Pour les symboles de nucléotides non ambiguës, il existe 99 triplets valides sur le total de 240 combinaisons hypothétiques, alors que 69 triplets valides existent dans le cas des symboles ambiguës. Les scores d'alignement par paires sont calculés pour toutes les paires possibles de triplets $(t_i, t_j) = ((\alpha_i, p_i, a_i), (\alpha_j, p_j, A_j))$ par une formule de *log-odds ratio* classique:

$$\sigma(t_i, t_j) = \lambda \log \frac{f_{t_i t_j}}{b_{t_i t_j}}. \tag{2}$$

Dans cette formule, $f_{t_i t_j}$ est la fréquence de la substitution $t_i \leftrightarrow t_j$ dans des séquences apparentées, et $b_{t_i t_j} = p(t_i)p(t_j)$ est la probabilité de retrouver cette paire de symboles dans des alignements aléatoires.

Les probabilités $f_{t_i t_j}$ sont définies par rapport au scénario de divergence présenté précédemment. Soit $p(c' \xrightarrow{\theta} c)$ la probabilité que le codon c' se transforme en c par mutations ponctuelles en temps évolutif θ, et $p_f^{I_i,I_j}(c_i, c_j)$ la probabilité que les codons c_i et c_j soient alignés avec un décalage de cadre de lecture f, et qu'ils se chevauchent dans leur alignement sur les intervalles définis par I_i et I_j. Les probabilités $p(c' \xrightarrow{\theta} c)$ sont données par un modèle de substitution des codons, alors que les valeurs de $p_f^{I_i,I_j}(c_i, c_j)$ peuvent être obtenues à partir des fréquences des codons π_c et des sous-séquences communes des codons sur des cadres de lecture

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

différents. Ensuite, les probabilités $f_{t_i t_j}$ sont données par:

$$f_{t_i t_j} = p((\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j)) = \sum_{\substack{c_i \text{encodes } a_i, \\ c_j \text{encodes } a_j}} \sum_{c_i', c_j' : c_i'[I_i] \equiv c_j'[I_j]} p(c_i'[I_i] \equiv c_j'[I_j]) \cdot p(c_i' \xrightarrow{\theta} c_i) \cdot p(c_j' \xrightarrow{\theta} c_j). \tag{3}$$

La probabilité $b_{t_i t_j}$ peut être exprimée comme la probabilité que les deux symboles t_i et t_j apparaissent de manière indépendante dans les séquences:

$$b_{t_i t_j} = b_{(\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j)} = \sum_{\substack{c_i \text{encodes } a_i, \\ c_j \text{encodes } a_j}} \pi_{c_i} \pi_{c_j}. \tag{4}$$

Expériences

L'approche est illustrée par quelques exemples (connus et nouveaux) d'alignements de protéines avec des *frameshifts*, dont certaines ne sont pas détectables par les méthodes classiques d'alignement, car la similarité de leur séquences codantes est très faible. Ces exemples soutiennent l'applicabilité de la méthode dans la découverte d'homologies lointaines.

Perspectives

Des futurs travaux concernent des raffinements du système de score, par exemple pour améliorer la détection des doubles changements de cadre de lecture séparées par un petit nombre de nucléotides, dont le deuxième changement corrige le déphasages créé par le premier. Des telles modifications se produisent assez fréquemment [175], mais sont souvent fortement pénalisés par les méthodes classiques de comparaison de séquences, qui rejettent l'alignement correct en faveur d'un autre alignement sans *gaps* et avec un score plus élevé.

Parmi les extensions possibles de ce travail, il serait intéressantde réaliser des alignements multiples de graphes de traduction inverse. Cette fonctionnalité pourrait être utile pour confirmer des *frameshifts* par similitude entre protéines qui sont issues d'une même famille.

Algorithmes d'alignement pour la technologie de séquençage SOLiD

Cette partie de la thèse propose des méthodes pour le *mapping* sur un génome de référence de *reads* obtenus par la plate-forme de séquençage à haut débit SOLiD.

La plate-forme SOLiD produit des *reads* de 35-50 paires de bases, représentés comme des séquences de 4 couleurs préfixés par un symbole de nucléotide qui permet leur décodage sans ambiguïté. Ces *reads* sont généralement alignés (*mappés*) sur une séquence génomique de référence qui est proche de celle qui a été séquencée, dans le but de découvrir des variations biologiques telles que des SNPs, des insertions/suppressions de nucléotides, ou même variations de structure. Le codage utilise 4 couleurs pour désigner les 16 paires possibles de nucléotides adjacents qui peuvent apparaître dans une séquence. En raison de l'utilisation du double codage des paires de nucléotides adjacents en couleurs, les positions d'un *read* ne peuvent pas être interprétées de manière indépendante, ce qui empêche les algorithmes d'alignement classiques d'exploiter la capacité de correction d'erreurs du code, et de capturer la similarité au niveau des séquences d'ADN par simple alignement de séquences de couleurs.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Le codage utilisant des paires de nucléotides adjacents est lié à la chimie de séquençage SOLiD. La plate-forme SOLiD utilise le séquençage par ligature, où deux nucléotides consécutifs sont interrogés tous les 5ème positions, en 5 étapes consistant en des cycles de plusieurs ligatures. Cela suggère que les erreurs de lecture peuvent avoir tendance à apparaître avec une périodicité de 5, une intuition confirmée par une analyse statistique effectuée sur les données de séquençage. En conséquence, une bonne compréhension de la plupart des types d'erreurs fréquentes et leur répartition sur le *read* peut améliorer sensiblement les résultats des outils de *read mapping*.

Conception de graines pour la technologie SOLiD

Une première contribution, présentée dans [71, 154, 155], consiste à concevoir des graines en se basant sur des modèles probabilistes qui modélisent à la fois les erreurs de lecture du séquenceur et les différences de nature biologique entre les *reads* et la séquence référence. La conception de graines utilise une approche formelle basée sur des automates finis, proposée dans [110]. Dans ce cadre, nous proposons un nouveau principe de graine spécialement adapté au *read mapping*, qui consiste à utiliser un petit nombre de graines *conçus conjointement avec un ensemble de positions sur le read*. Ces *graines positionnées* permettent de prendre en compte des propriétés telles que la distribution non-uniforme des erreurs de lecture sur le *read*, ou bien de mieux prendre en compte leur tendance à se produire plus régulièrement à des distances de 5 positions. Des graines espacées *avec perte* et *sans perte* peuvent être conçues dans ce cadre.

Conception de graines avec perte Il existe deux sources indépendantes créant des *différences* entre des *reads* et un génome de référence: d'une part les erreurs de lecture du séquenceur, et d'autre part les variations de séquence (SNPs, insertions/suppressions que l'on note aussi *indels*). Nous représentons chacune de ces sources par un Modèle de Markov Caché (Figure 2 et Figure 3). Les deux modèles sont ensuite combinés (par produit d'automates) dans un modèle qui permet de cumuler tous les types d'erreurs dans les séquences résultantes, tel qu'il est montré dans l'exemple de la Figure .

La sensibilité d'une famille de graines est ensuite calculée l'aide de la technique de programmation dynamique proposée dans [110] sur le produit de l'automate spécifiant une famille de graines avec le modèle d'alignement cible construit comme expliqué ci-dessus.

Conception de graines sans perte Nous proposons un algorithme de programmation dynamique efficace appliqué directement à l'automate $\mathcal{Q}$, correspondant à une graine (ou à une famille de graines) donné. Cet algorithme peut vérifier si la graine est *sans perte* par rapport aux alignements de taille m avec un certain nombre d'erreurs k : elle est capable de détecter tous les alignements de taille m ayant k erreurs. Cet algorithme calcule, pour chaque état q de $\mathcal{Q}$, et pour chaque itération $i \in [1..m]$, le nombre minimal de changements de couleurs (substitutions) nécessaires pour atteindre tout état q à l'étape i. La condition "sans perte pour k erreurs" est vraie si et seulement si, à l'étape m, tous les états non-finaux ont un nombre de substitutions supérieur à k. En effet, s'il existe un état non-final avec un nombre d'erreurs inférieur ou égal à k après m itérations, alors il y a au moins une chaîne de longueur m avec moins de k (ou k) substitutions qui n'est pas acceptée par l'automate, ce qui contredit la condition sans perte. Cet algorithme est de complexité temporelle $\mathcal{O}(|\mathcal{Q}| \cdot |\mathcal{A}| \cdot m)$, et de complexité en espace $\mathcal{O}(|\mathcal{Q}| \cdot |\mathcal{A}|)$, avec $\mathcal{A}$ définissant l'alphabet des alignements cible.

Dans le cadre des alignements de séquences de couleurs, il est intéressant de définir la propriété sans perte par rapport à une certaine combinaison d'erreurs de lecture et substitutions de nucléotides (SNPs). Un exemple d'automate qui reconnaît l'ensemble des alignements avec

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

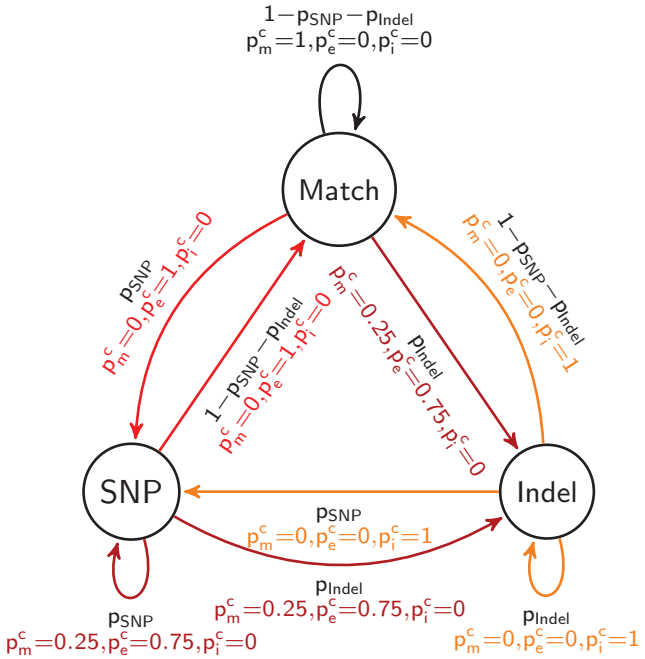

Figure 2: Modèle de Markov des variations biologiques. Les états décrivent l'alignement au niveau de l'ADN, et les transitions décrivent l'alignement au niveau du code couleur. Les couleurs des transitions correspondent aux émissions: en noir pour les paires de couleurs identiques, en rouge pour les substitutions de couleurs, en jaune pour les insertions/suppressions, en rouge foncé pour un mélange de substitutions et conservations.

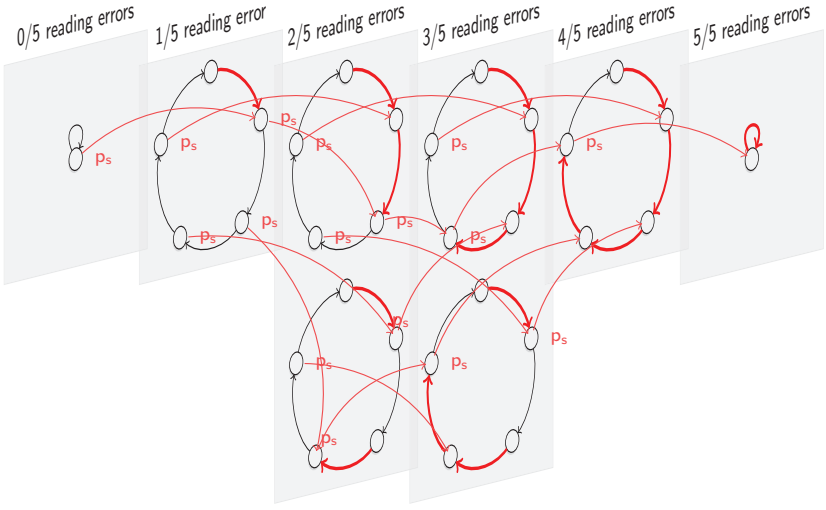

Figure 3: Modèle de Markov Caché des erreurs de lecture.

http://doc.univ-lille1.fr

$$
\begin{array}{cc}
M_{SNP/I} & 1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,0\,\mathtt{I}\,1\,1\,1\,1\,1\,\mathtt{I}\,1\,1\,1\,1\,1\,1 \\
\times & \\
M_{RE} & 1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0 \\
= & \\
M_{(SNP/I)\times RE} & 1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,0\,\mathtt{I}\,1\,1\,1\,1\,0\,\mathtt{I}\,0\,1\,0\,1\,0\,0
\end{array}
$$

Figure 4: Le modèle d'alignement qui combine les deux sources d'erreurs (variations biologiques et erreurs de lecture) génère des alignements où tous les types d'erreurs sont cumulés. Ici, tt 1 est désigne une paire de couleurs identiques alignées, 0 représente une substitution de couleur, et I représente une insertion dans le *read*.

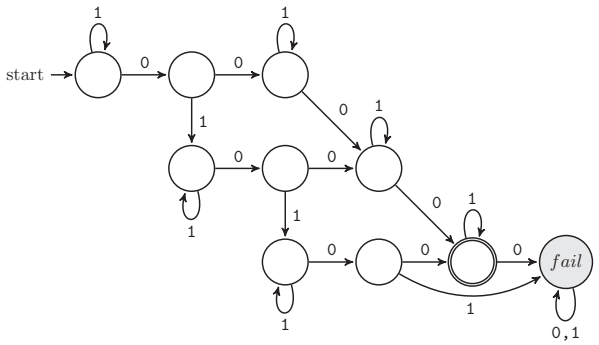

Figure 5: Automate qui modélise les alignements avec 1 SNP & 2 erreurs de lecture.

1 SNP et 2 erreurs de lecture est illustré dans la Figure 5. La construction de cet automate s'appuie sur le fait que, dans l'espace de couleurs, un SNP apparaît en fait comme deux substitutions de couleurs consécutives. Dans une approche similaire, des automates plus complexes qui tiennent compte des insertions et suppressions peuvent être conçus.

Pour vérifier si une graine est sans perte par rapport aux alignements avec une certaine combinaison d'erreurs, il suffit de "croiser" l'automate qui modélise ces alignements avec l'automate de la graine (limitant ainsi l'ensemble des alignements reconnus par la graine à ceux ayant la combinaison d'erreurs établie) et de soumettre ensuite le résultat à l'algorithme de programmation dynamique décrit ci-dessus.

Algorithme d'alignement pour des séquences de couleurs

L'objectif de cet algorithme est d'obtenir des alignements de séquences de couleurs qui sont significatifs au niveau de la séquence des nucléotides, c'est à dire de correctement faire la distinction entre les erreurs de lecture et les variations biologiques. L'algorithme effectue un alignement dit en *bandwidth*, dont la largeur est ajustée pour tenir compte du nombre maximum autorisé d'insertions/suppressions. Il est basé sur l'approche classique d'alignement semi-global, enrichi d'une *mémoire limitée des décalages de couleurs* sur chaque chemin (alignement partiel). En bref, tout décalage de couleur qui n'est pas "corrigé" (suivi par d'autres décalages qui mèneront éventuellement à la même base dans les deux séquences de nucléotides) après un certain nombre de paires de couleurs est considéré comme une erreur de lecture.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Expériences

Les concepts présentés ci-dessus ont été mis en œuvre dans un outil expérimental de *read mapping*. Les expériences montrent la compétitivité des graines conçues avec notre approche par rapport aux autres logiciels de *read mapping* existants.

Perspectives

L'objectif principal de ce travail est de proposer un cadre de conception de graines adaptées à l'alignement des *reads* SOLiD avec une séquence génomique de référence. Les contributions comprennent le concept de *graine positionnée*, particulièrement adapté pour des alignements courts avec une distribution d'erreurs non-uniforme; un modèle qui capture les caractéristiques statistiques des *reads* SOLiD, utilisé pour l'évaluation des graines avec perte; un algorithme efficace de programmation dynamique pour vérifier si une graine ou une famille de graines est sans perte; la capacité à distinguer les SNPs, les erreurs de lecture et les *indels* dans la conception des graines sans perte; un algorithme pour l'alignement des séquences de couleurs, qui donne la possibilité de distinguer les mutations et les erreurs de lecture; un outil expérimental de *read mapping* qui met en œuvre ces concepts.

De futurs travaux comprendront une estimation plus précise des probabilités utilisées dans les modèles. En outre, le logiciel IEDERA peut bénéficier d'une extension permettant de concevoir des familles de graines pour des stratégies de *hit* multiple. Une autre question intéressante à étudier est la conception de graines efficaces qui sont à la fois *avec perte* et *sans perte*, donc qui garantissent de trouver tous les alignements avec un certain nombre d'erreurs et qui continuent à avoir une bonne sensibilité lorsque ce seuil est dépassé. Toutefois, puisque les graines *sans perte* ont tendance à avoir une structure régulière [109], tandis que les meilleures graines *avec perte* sont souvent asymétriques et irrégulières, obtenir telles graines peut être difficile.

Conclusions

Cette thèse porte sur la conception d'algorithmes d'alignement exacts et heuristiques pour répondre à deux problèmes complexes de similarité de séquences: i) la détection d'homologies cachés entre les protéines *par comparaison de séquences protéiques*, lorsque la source de la divergence est dû à des mutations qui changent le cadre de lecture, et ii) le *mapping* de *reads* SOLiD (séquences de di-nucléotides chevauchants codés par des couleurs) sur un génome de référence.

L'approche proposée pour le premier problème est basée sur des modèles d'évolution de codons. Les origines communes des protéines sont recherchées par alignement implicite de toutes leur séquences d'ADN possibles, stockées efficacement dans des graphes de traduction inverse. Les propriétés de triplet de l'ADN codant sont capturées dans l'algorithme de programmation dynamique conçu pour aligner ces graphes et dans son système de scores. Contrairement aux algorithmes classiques d'alignement de séquences, cette approche ne fait pas l'hypothèse d'indépendance entre les paires de symboles alignés. Le système de scores prend en compte la traduction des séquences alignées, afin d'obtenir des alignements qui reflètent la dynamique des séquences codantes.

Pour le deuxième problème, les solutions algorithmiques proposées intègrent les connaissances sur les caractéristiques de l'encodage SOLiD et sur les artefacts de cette technologie de séquençage. Ces deux aspects sont capturés par des modèles probabilistes. Le double codage des nucléotides en couleurs adjacentes est géré par l'interprétation implicite des séquences de couleurs comme des séquences ADN, en tenant également compte des erreurs de lecture possibles.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

L'accent est principalement mis sur la conception de graines espacées efficaces qui permettent d'identifier les positions candidates des *reads* sur le génome de référence. Cette étape a des effets importants sur la qualité et la vitesse du processus de *read mapping*.

En conclusion, dans les deux cas, la même idée générale est appliquée: la comparaison implicite des séquences d'ADN pour la détection de changements qui se produisent à ce niveau, lors de la manipulation, en pratique, d'autres représentations (séquences de protéines, des séquences de codes di-nucléotides). Ces autres représentations fournissent à chaque fois des informations supplémentaires et aident à améliorer la recherche de similarités. La thèse propose des méthodes génériques pour résoudre les problèmes abordés, ainsi que des implémentassions pour ces méthodes, librement disponibles pour l'utilisation en ligne ou en téléchargement.

Ce travail ouvre de nouvelles perspectives dans la conception de modèles et d'algorithmes adaptés aux problèmes spécifiques d'alignement de séquences, et soutient l'idée de comprendre et d'utiliser les propriétés statistiques des séquences alignées, en les capturant dans des algorithmes d'alignement dédiés, des fonctions de scores associées, et des concepts de graines adaptés à ces propriétés.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Introduction

Biological sequence alignment is a fundamental technique in bioinformatics, and consists of identifying series of similar (conserved) characters that appear in the same order in both sequences, and eventually deducing a set of modifications (substitutions, insertions and deletions) involved in the transformation of one sequence into another. This technique allows one to infer, based on sequence similarity, if two or more biological sequences are potentially homologous, i.e. if they share a common ancestor, thus enabling the understanding of sequence evolution.

From a computational perspective, sequence comparison methods are strongly related to string editing, since biological sequences are usually represented as strings over a certain alphabet: the alphabet of the four *nucleotides* for DNA sequences, or the alphabet of the twenty *amino acids* for proteins. The relatedness of the aligned sequences is quantified by the *score* assigned to their alignment, which comprises the scores associated (usually independently) to individual pairs of aligned characters, reflecting their similarity. Algorithms generally employ a *dynamic programming* approach in order to obtain the best sequence alignment with respect to a certain scoring scheme. The first such algorithms [75, 150, 188] and scoring systems [51, 86] were proposed several decades ago, and continue to be used in practice. Later on, the emergence of DNA sequencing technologies [170,176,181] led to impressive amounts of data becoming available and needing to be analyzed. Thus, in complement to exact alignment algorithms, fast heuristic approaches [11,130,164] have been developed in order to cope with very large sequences whose exact alignment would require an unreasonable computation time. These heuristics are based on the assumption that sufficiently related sequences must share at least a well conserved region, called *seed*. The approach basically consists of identifying that region if it exists, and then constructing an optimal alignment around it.

This generic algorithmic recipe is suited for most classic problems of protein or DNA sequence comparison. Nevertheless, with the increasing amounts of available data and the wide range of new applications, specific problems have emerged that may require many aspects of the general framework to be adapted to their characteristics. Various approaches subsequently developed have shown that for instance, the procedures of handling pairs of aligned symbols and gaps within the alignment algorithm [15, 16, 39, 79, 85, 106, 113, 166, 168, 214], the scoring scheme [40, 65, 167], or the similarity patterns considered "relevant" in seeded alignments [26, 216] can be adjusted to incorporate knowledge about a certain problem and thus capture the degree of similarity between the compared structures. Moreover, much attention has been channeled towards the actual *design* of similarity patterns sought by seeds in order to comply with models capturing the characteristics of the aligned sequences [30, 31, 99, 105, 110, 133].

This thesis addresses sequence comparison problems in two different contexts: homology detection and high throughput DNA sequencing.

The use of sequence comparison as an instrument for homology detection relies on the known fact that gene duplication followed by divergence plays an impressive role in evolution [4].

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

"Natural selection merely modified, while redundancy created" [158] is a hypothesis formulated decades ago in an attempt to explain major advances in evolution, such as the transition from single cell life forms to complex multicellular organisms [145]. Later on, studies on completely or nearly completely sequenced genomes from a wide range of organisms have shown that a large proportion of genes have indeed been generated via duplication, while the origin of others is thought to reside in this mechanism, but the proof is hindered by their advanced divergence [213]. As such, while sequence similarity is considered to be strong evidence of homology, it does not guarantee homology if present, and conversely sequences that seem dissimilar are not necessarily unrelated, hence *the need for comparison methods powerful enough to reveal sequence similarities that are actually relevant to homology.*

Sequence divergence after duplication is the result of various types of *mutations*, such as substitutions, insertions, deletions of nucleotides. If they affect a coding sequence, i.e. a piece of DNA that encodes a protein with a certain function within the organism, these modifications implicitly influence the product of that sequence, namely the protein translated from it, and may change its function or make it nonfunctional. A particularly interesting homology detection problem consists in finding similarities between proteins having diverged as result of a frameshift. Frameshifts are insertions or deletions of several nucleotides in a coding DNA sequence. A single frameshift mutation introduces a drastic change in the translated protein sequence, completely changing the translation after the frameshift, while having a small effect on the affected DNA sequence, most of which is left unchanged. However, if additional nucleotide substitutions are further involved in the divergence, the similarity at the DNA level may be reduced beyond recognition. With such important changes occurring in the DNA sequences and their translations, classic alignment algorithms are likely to fail at detecting the homology, both at the DNA and protein sequence level. Algorithmic solutions for this problem are proposed in **Part II** of the thesis.

Homology detection is not the only context where biological sequence analysis relies on sequence comparison. Alignment methods are a key ingredient in *assembling* or *mapping onto a reference genome* short "reads" produced by high throughput sequencing technologies, representing pieces of the target sequence that can range from tens to hundreds of symbols. In the context of read mapping, one of the challenges resides in the identification of the candidate mapping position on the reference sequence for each read, which basically requires fast detection of relevant similarities between the read and the reference. The identification, as well as the alignment itself of the read onto the reference sequence, must cope with possible sequencing errors and technology artifacts that may alter the reads, in addition to biological variations.

One particular sequencing technology, the SOLiD platform [1], features a 2-base encoding with an error-correcting capability with the purpose of *improving* the distinction between sequencing errors and biological variations. Basically, instead of being a sequence of nucleotide symbols, a SOLiD read is a sequence of four possible *colors*, each color encoding two consecutive nucleotides and each nucleotide being comprised in two consecutive colors in the sequence. Because of the dually encoded nucleotide information in adjacent colors, read positions cannot be interpreted independently, which prevents classic pairwise alignment algorithms from capturing the DNA sequence similarities by simple color sequence alignment, since classic alignment methods generally handle pairs of aligned symbols independently. Instead, biological modifications and reading errors can be recognized and distinguished by analyzing several consecutive color positions simultaneously. This problem is approached in **Part III** of the thesis, which discusses methods adapted to its characteristics.

To sum up, the goal of this work is to develop sensitive alignment methods that provide

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

solutions to the following two problems: i) the detection of hidden protein homologies *by protein sequence comparison*, when the source of the divergence are frameshift mutations, and ii) mapping short SOLiD reads (sequences of overlapping di-nucleotides encoded as colors) to a reference genome. *In both cases, the same general idea is applied: to implicitly compare DNA sequences for detecting changes occurring at this level, while manipulating, in practice, other representations (protein sequences, sequences of di-nucleotide codes) that provide additional information and thus help to improve the similarity search.* The aim is to design and implement exact and heuristic alignment methods, along with scoring schemes and evaluation techniques, adapted to these scenarios.

Structure of this thesis

Part I of this thesis gives the state of the art, introducing biological concepts and algorithmic methods that constitute the basis of the work presented in the rest of the thesis.

As such, **Chapter 1** presents the biological entities involved in the problems addressed by this thesis. It first presents nucleic acids (DNA and RNA) and proteins, with their respective physical and chemical properties and their roles within the organisms. It further gives an overview of the central dogma of molecular biology, which is fundamental for understanding the flow of information from coding sequences (DNA, RNA) to those ensuring the functioning of the organism (proteins). A section on the subject of mutations, which are at the origin of evolution and whose discovery is the target of sequence comparison methods, concludes this chapter.

Chapter 2 gives an overview of sequencing technologies which allow "reading" the genome and produce data for its computational analysis. The DNA replication presented briefly in the previous chapter in the context of the central dogma of molecular biology creates a foundation for understanding the chemistry behind sequencing.

Chapter 3 gives a computational perspective of sequence comparison. First, classic algorithms for pairwise sequence alignment are presented, followed by an overview of existing scoring systems for sequence comparison, and the methods employed for their creation. The final part of this chapter focuses on heuristic methods based on seeds for detecting similarities in large sequence databases, and discusses the design of spaced seeds adapted for a specific sequence comparison problems.

Part II addresses the problem of *finding hidden protein homologies whose divergence is caused by a frameshift, using direct protein comparison.*

First, **Chapter 4** introduces the problem, gives a related work overview and summarizes the contributions of this part.

The proposed alignment method is presented in **Chapter 5**. After describing the concept of *protein back-translation* on which the approach relies, and introducing the representation of proteins as *back-translation graphs*, a new *dynamic programming algorithm for aligning two such graphs* is proposed. Within this algorithm, frameshifts are handled via a non-monotonic gap penalty function, where insertions and deletions of full codons are penalized less than reading frame disruptive gaps. Additionally, since frameshifts are considered to be very rare events, the algorithm offers the possibility to restrict their number in an alignment. This algorithm is used in conjunction with a *powerful scoring system* designed to reflect the actual evolution process from a codon-oriented perspective.

To facilitate large scale analysis, **Chapter 6** proposes seeding techniques for the search of protein similarities when frameshifts are involved in their divergence.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

The approach is illustrated in **Chapter 7** with several alignment examples of known and hypothetical frameshifted proteins, some of which are not detectable via classic alignment methods because of the low coding sequence similarity.

Part III proposes solutions for *mapping to a reference genome short reads obtained by next generation sequencing technologies*, and focuses on the SOLiD platform in particular.

Chapter 8 first describes the SOLiD encoding and presents the observed artifacts of the platform, in order to explain the difficulties encountered in the context of mapping SOLiD reads, and continues with overview of existing approaches (algorithms and tools). The contributions are then announced in a summary of the following three chapters.

Chapter 9 focuses on a seed design framework based on Hidden Markov Models of read matches, using a formal finite automata-based approach previously developed in [110]. The framework allows the design of spaced seeds, both lossless and lossy, based on a *probabilistic model of SOLiD read alignments* incorporating reading errors, SNPs and base indels, A *new seeding principle* especially adapted for read mapping is proposed, relying on the use of a small number of seeds *designed simultaneously with a set of position on the read where they can hit.* This principle of *positioned seeds* allows taking into account, in a subtle way, read properties such as a non-uniform distribution of reading errors along the read, or a tendency of reading errors to occur periodically, which are observed artifacts of the SOLiD technology.

Chapter 10 discusses a "base-intelligent" alignment method designed for color sequences, which makes use of the encoding's error-correcting properties and implicitly takes into account the similarity at the DNA sequence level, thus improving the distinction between reading errors and biological variations.

Finally, **Chapter 11** presents an experimental read mapping tool designed for SOLiD which implements these idea, discussed in comparison with several existing read mapping applications.

The thesis **concludes** with a brief overview of the contributions and perspectives.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Part I

State of the art

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 1

Biological context

This chapter presents the biological context of the problems approached in the thesis. It introduces notions related to the biological entities and processes that are the object of our work. Hence, we begin by overviewing the characteristics of information-carrying biopolymers, with focus on DNA and proteins. We then proceed to present the basics of the central dogma of molecular biology, which aims at explaining the information transfer between DNA, RNA and protein sequences through the transcription and translation processes. Finally, we discuss different types of mutations and their possible effects on the functioning of the affected organism.

1.1 Nucleic acids

1.1.1 The deoxyribonucleic acid (DNA)

Role within the organism

DNA is a macromolecule forming the *genome* of living organisms. It can be considered the source code of life, because it stores the genetic information necessary for development, functioning and reproduction. The DNA is found in the cytoplasm of prokaryotic organism cells, whereas in eukaryotic organisms it is contained in majority inside the cell nucleus, and in small proportions inside organelles such as mitochondria. In multicellular eukaryotes, each nucleated cell has a full copy of the genome, since DNA possesses the genetic information that ensures heredity and thus it is essential for cell reproduction.

DNA is organized in chromosomes. A chromosome consists of a single, very long DNA molecule that contains a linear array of many genes, regulatory elements and other nucleotide sequences, along with DNA-bound proteins, which serve to package the DNA in a compact form and control its functions. Genes are the DNA regions that carry the genetic information. In addition to these, DNA has other regions with structural purposes, or involved in regulating the expression of genes. Different variants of a gene that can appear at a given locus (specific location on a chromosome) are called alleles [5]. While the DNA of prokaryotes has relatively few non-coding regions, in more complex organisms the majority of the DNA remains without a clearly identified function. As an example, approximately 2% of the human genome is actually protein-coding [178].

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Figure 1.1: DNA double helix (schematic representation).

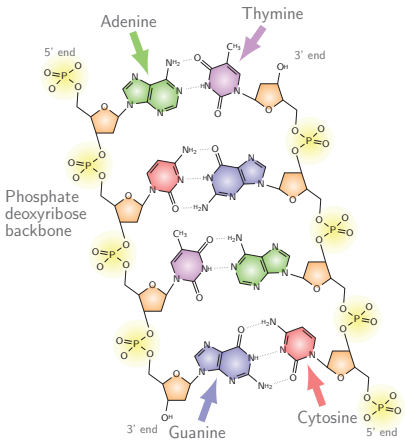

Figure 1.2: The chemical structure of the DNA.

Chemical structure

DNA is usually found as a double strand, forming the famous double helix model (Figure 1.1) that was published by James Watson and Francis Crick in 1953 [207], as evidenced in the X-ray diffraction images taken in 1952 by Rosalind Franklin and Raymond Gosling [63].

Figure 1.2 zooms in on the chemical structure and gives a schematic representation at the atomic level of a small DNA segment (4 base pairs), in order to illustrate the following details concerning the actual composition.

The DNA consists of two long polymers of units called *nucleotides*. The two polymers have backbones made of five-carbon sugars (deoxyribose) linked by phosphate groups, and each sugar is attached to one of four types of molecules called *nucleobases*, or simply *bases* (Figure 1.3): adenine (A), guanine (G), cytosine (C) and thymine (T). The nucleotide is composed of one base, one sugar (together forming a compound called a *nucleoside*) and one phosphate [120]. Adenine and guanine are known as *purines*, and resemble in that they have a double ring in their chemical structure, while cytosine and thymine are called *pyrimidines*, and have a single ring [87]. **It is the succession of these bases in the DNA sequence that encodes the entire genetic information of a living organism**.

The carbons in the sugar are conventionally numbered from 1 to 5, and the phosphate groups are linked to carbons 3 and 5 (Figure 1.5). At one end of the polymer, the last carbon in the chain is a number 5 carbon, and at the other end, the last carbon is a number 3, thus the names *5' end* and *3' end*. It is considered that a strand begins at 5' and ends at 3', to comply with the direction of reading the genetic information [87].

The two polymers (strands) are held together by *hydrogen bonds* between the bases A and T and between C and G (Figure 1.4). The paired bases are called *complementary* (A is the complementary of T and forms with it 3 hydrogen bonds, and C is the complementary of G, forming two hydrogen bonds with it). The two strands have completely complementary sequences and opposite directions (the 5' end of one strand corresponds to the 3' end of the other, i.e. they are *antiparallel*) and they are coiled around one another in the shape of a *double helix* [207].

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

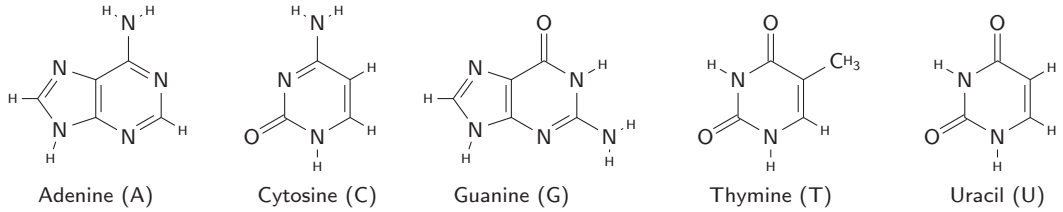

Figure 1.3: The five bases that can be found in DNA and RNA: adenine, cytosine, guanine (DNA and RNA), thymine (DNA), uracil (RNA).

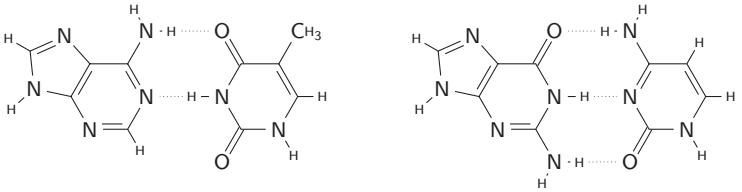

Figure 1.4: Hydrogen bonds (dashed lines) between complementary bases: adenine pairs with thymine (A-T), and guanine with cytosine (G-C).

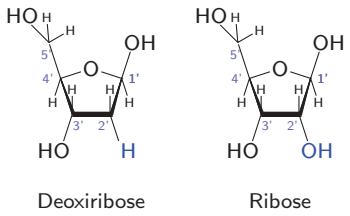

Figure 1.5: The *deoxiribose* (left) and *ribose* (right) sugars, contributing to the backbone of DNA and RNA respectively.

1.1.2 The ribonucleic acid (RNA)

Also a polymer of nucleotide units, RNA differs from DNA in both structure and function in several ways. First, RNA is usually a single-stranded, much shorter molecule. Secondly, the sugar contained in the RNA nucleotides is a ribose (similar to the deoxyribose sugar from the deoxynucleotides forming the DNA, with an extra hydroxyl group attached to carbon number 2, see Figure 1.5), which makes RNA less stable than DNA because it is more prone to hydrolysis [19]. Finally, the complementary base to adenine is not thymine, as in DNA, but another pyrimidine called uracil (U) [19] illustrated in Figure 1.3, and some RNAs (such as tRNA) may contain a variety of modified bases, especially by methylation [56, 189].

Function-wise, there are several known types of RNA, involved in cellular processes such as regulation, translation, RNA processing. Two of these, *messenger RNA (mRNA)* and *transfer RNA (tRNA)*, will be explained in Section 1.3 in the context of protein synthesis.

1.2 Proteins

Role within the organism

Proteins play an essential part in the functioning of an organism. They are actively involved in the vast majority of cellular processes [131]: biochemical reaction catalysis (*enzymes*), *transport* and *storage* of nutrients and other molecules, regulatory functions (*hormones*), defense (*antibodies*), providing *structural* rigidity to the cell (strengthening protective coverings such as hair, feathers, horns, and beaks, or providing support for connective tissues such as tendons and

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

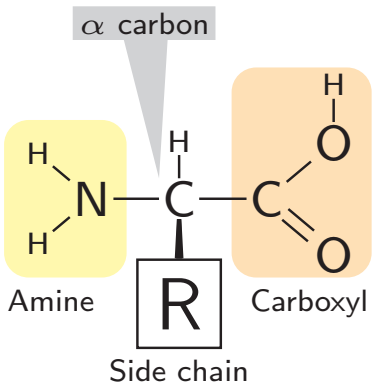

Figure 1.6: The general structure of an amino acid.

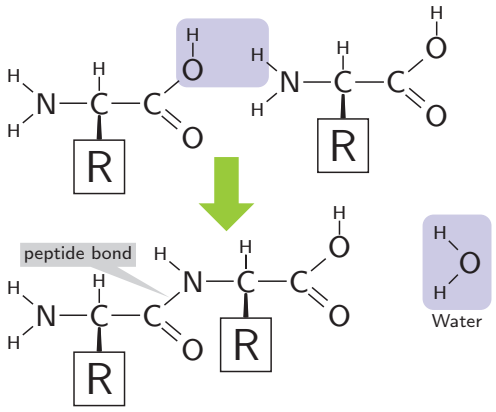

Figure 1.7: The condensation of two amino acids to form a peptide bond.

ligaments) or causing motion (such as muscle contraction).

The function of a protein is determined by its shape in three dimensions, which depends on its chemical composition and is influenced by environmental factors [20, 131].

Chemical structure

A protein is a chain of (usually several hundreds of) units called *amino acids*. These amino acids are organic compounds having an amine group (NH_2) and a carboxylic acid (COOH) linked to a carbon atom called *the α carbon* [20]. Their general structure is depicted in Figure 1.6. The amino acids are chained via a peptide bond, formed by removing an OH from one amino acid and a H from the next, with the release of a water molecule [20], as illustrated in Figure 1.7. At the two ends of a protein, there remain an unlinked amine (NH_2) and an unlinked carboxyl (COOH) respectively, hence the names *N terminus* and *C terminus*. The order of amino acids in the chain is considered conventionally from the N terminus to the C terminus, to comply with the sense of the protein's synthesis [87] (see Section 1.3).

The proteins of most organisms are composed of the 20 amino acids shown in Figure 1.8. In some eukaryote proteins, a 21st amino acid called *Selenocytosine* may appear [90]. The amino acids differ by the nature of their side-chain group, denoted **R** in Figure 1.6, and attached to the α carbon. **The succession of amino acids in the chain defines the protein sequence** (also known as *primary structure*).

The 4 atoms involved in the peptide bond are located in the same plane and cannot rotate relatively to one another. Thus, the shape of the protein backbone is mostly given by the bonds involving the α carbon [87].

The protein's *secondary structure* (Figure 1.9) is given by the general three-dimensional form of local segments, defined by patterns of hydrogen bonds between backbone amide and carboxyl groups. The most common secondary structures are α helices (in which every backbone NH group donates a hydrogen bond to the backbone CO group of the amino acid four positions earlier) and β sheets (consisting of two parallel or antiparallel strands in which the NH groups in the backbones of one strand establish hydrogen bonds with the CO groups in the backbone of the adjacent strand). The *tertiary structure* is defined by specific atomic positions in three-dimensional space, and finally the *quaternary structure* refers to the arrangement of multiple folded macromolecules in a complex [20].

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

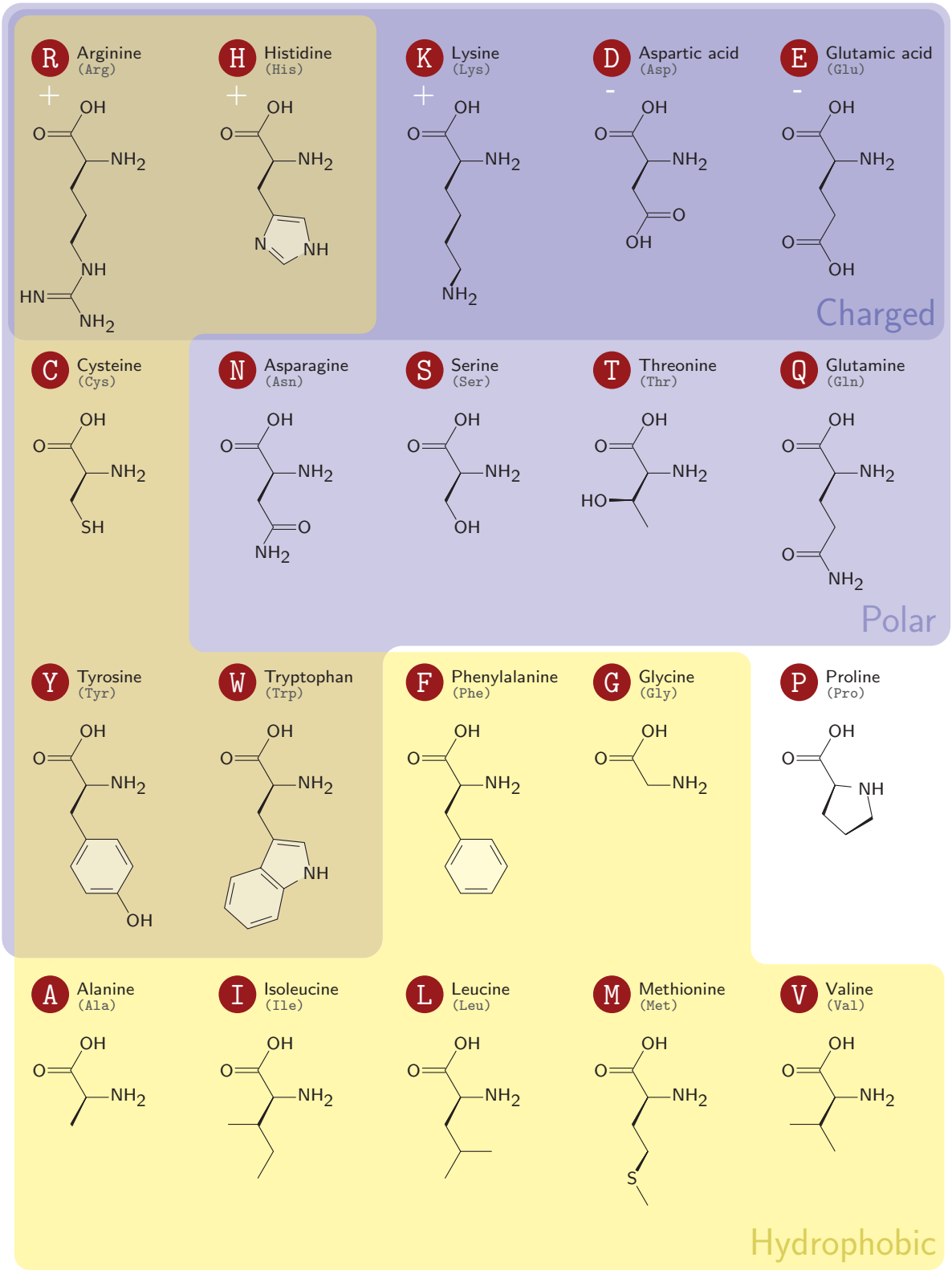

Figure 1.8: The twenty amino acids, grouped according to their side chains' polarity and charge.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

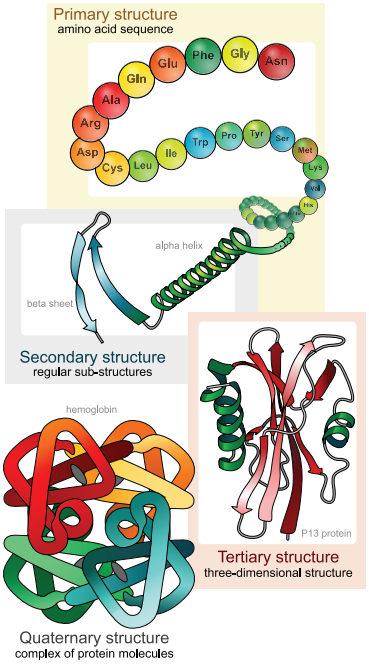

Image source:

`http://en.wikipedia.org/wiki/File:Main_protein_structure_levels_en.svg`

Figure 1.9: Protein structure levels.

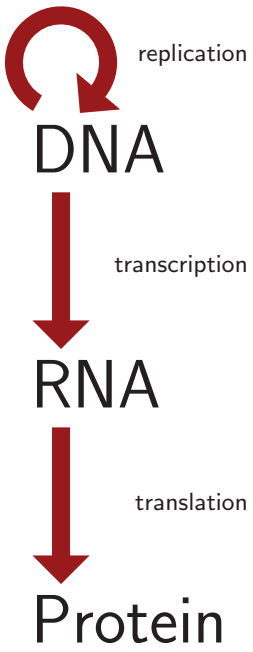

Figure 1.10: Central Dogma of Molecular Biology: from DNA through RNA (transcription) to proteins (translation).

In the following (Part II), we will not discuss protein folding and interactions. We will mainly be interested in the protein's primary structure, since we search for similarities at the sequence level. A protein's sequence of amino acids is encoded in the genome by its corresponding gene, and is synthesized as explained in Section 1.3.

1.3 Central dogma of molecular biology

The central dogma of molecular biology, first stated by Francis Crick in 1958 and published in 1970 [45], aims at explaining the transfer of sequence information between biopolymers (nucleic acids and proteins) in living organisms.

Basically, the dogma states that DNA acts as a template to *replicate* itself, DNA is also *transcribed* into RNA, and RNA is *translated* into protein. The information transfer is schematically illustrated in Figure 1.10.

1.3.1 Transcription

As mentioned in Section 1.1.1, *genes* are regions of DNA that contain the information necessary for defining the primary structure of RNA and proteins.

The synthesis of an RNA copy of a coding DNA region is called *transcription*, and it is performed by an enzyme called *RNA polymerase*. During transcription, the two DNA strands are temporarily separated. The polymerase binds to one of the strands, called *template strand*, and, sliding from its 3' end towards the 5' end, it catalyzes the chaining of ribonucleotides in a

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

RNA strand that is complementary to the template, and therefore equivalent to the non-template strand of the DNA being transcribed [6].

Splicing

The RNA resulting from the transcription is called *messenger RNA (mRNA)*, and contains the information necessary to produce the protein, along with some regions that will not be translated. In prokaryotes these untranslated regions are rather short and located only at the ends of the RNA sequence. In eukaryotes the composition of the newly transcribed RNA (called *pre-mRNA*) is more complex and requires additional processing before translation. Eukaryotic gene sequences are composed of alternating regions: *exons*, holding the information for protein coding, and *introns* that do not contain protein-coding information and will not be translated [7]. The removal of introns from the RNA in order to obtain a contiguous sequence formed only from exons is called *splicing* and is performed by the *spliceosome*, a complex of several types of RNA and proteins [88].

In eukaryotes, both the transcription and the RNA processing take place in the cell nucleus. Then, resulting mRNA is then transported out of the nucleus, and its translation is done in the cytoplasm.

1.3.2 Translation

The genetic code

The mRNA, a sequence of 4 possible types of nucleotides, contains the information for producing a protein, a chain formed with (usually) 20 possible types of amino acids. To build the protein sequence using mRNA as a template, the mRNA is read from its 5' end to the 3' end, in groups of 3 bases called *codons*, and each codon is interpreted as an amino acid [46] (Figure 1.11). The contiguous and non-overlapping set of 3 nucleotide codons is known as *reading frame*. Since there are $4^3 = 64$ different codons that can be obtained with the four nucleotides, and 20 amino acids, this encoding is redundant, and at least some of the amino acids have several corresponding codons. However, the code is not ambiguous: a codon cannot encode more than one amino acid.

The general rules of association between codons and amino acids are known as *the genetic code* [44,46,138,152,153], given in Figure 1.12. In this standard genetic code, 61 of the 64 codons encode amino acids, and the remaining 3 codons, namely UAA, UAG and UGA, are called *stop codons* and mark the termination of a coding sequence (translation is not carried beyond this marker). The codon AUG, encoding the amino acid *Methionine*, is also known as the *start codon*, since it is the point where translation is normally initiated. Out of the 20 amino acids, two are encoded by a single codon (*Methionine* and *Tryptophan*), 9 have 2 corresponding codons, one (*Isoleucine*) can be obtained from 3 different encodings, 5 amino acids have 4 codons each, and 3 (*Arginine*, *Leucine* and *Serine*) are encoded by 6 different codons. It can be observed that the codons corresponding to the same amino acid are very similar, and usually they have different bases only on their 3rd position. This rule is valid for all amino acids encoded by 2, 3 or 4 codons, but does not fully apply to those that are encoded by 6 codons, although each of them has two subsets of 4 and 2 codons respectively that have two identical positions. Thus, the codon sets of *Arginine* and *Leucine* also present differences on their first position, and *Serine* has encodings that differ on all three positions.

When initially discovered, this code was considered universal [44]. Later on, it was found that this code is extremely common, but not completely universal. It applies to almost all prokaryote

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

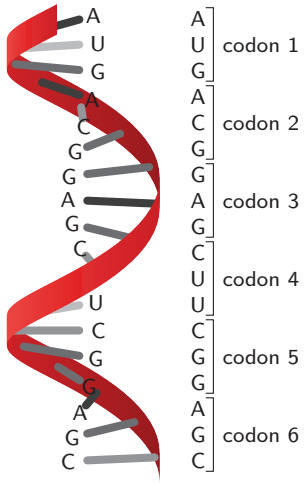

Figure 1.11: mRNA translation: bases are read in groups of 3, called *codons*.

UUU, UUC: F Phenilalanine (Phe) UUA, UUG: L Leucine (Leu)	UCU, UCC, UCA, UCG: S Serine (Ser)	UAU, UAC: Y Tyrosine (Tyr) UAA: stop codon UAG: stop codon	UGU, UGC: C Cysteine (Cys) UGA: stop codon UGG: W Tryptophan (Trp)
CUU, CUC, CUA, CUG: L Leucine (Leu)	CCU, CCC, CCA, CCG: P Proline (Pro)	CAU, CAC: H Histidine (His) CAA, CAG: Q Glutamine (Gln)	CGU, CGC, CGA, CGG: R Arginine (Arg)
AUU, AUC, AUA: I Isoleucine (Ile) AUG: M Methionine (Met)	ACU, ACC, ACA, ACG: T Threonine (Thr)	AAU, AAC: N Asparagine (Asn) AAA, AAG: K Lysine (Lys)	AGU, AGC: S Serine (Ser) AGA, AGG: R Arginine (Arg)
GUU, GUC, GUA, GUG: V Valine (Val)	GCU, GCC, GCA, GCG: A Alanine (Ala)	GAU, GAC: D Aspartic Acid (Asp) GAA, GAG: E Glutamic acid (Glu)	GGU, GGC, GGA, GGG: G Glycine (Gly)

Figure 1.12: The standard genetic code.

genomes and to the nuclear genomes of most eukaryotes. Different interpretations of several codons were found in mitochondrial genomes [103] and some eukaryotes [57]. For example, alternative start codons, such as GUG or UUG, have been discovered in some mammalian species [201], as well as in bacteria and archaea [57]. Also, in certain proteins, stop codons are interpreted as non-standard amino acids, as is the case of *Selenocytosine* being translated from the codon UGA [57, 90].

Protein synthesis

Proteins are synthesized by *ribosomes* using mRNA as a template. The ribosome is a complex of RNAs and proteins, and has two subunits: a small subunit that binds to the mRNA, and a large subunit that binds to the *transfer RNA (tRNA)* which carry the amino acids that will compose the resulting protein. The transfer RNA (tRNA) is folded in a clover shape (Figure 1.13), and its key element is a sequence of 3 bases called an *anticodon*. Each tRNA contains a specific anticodon that can form complementary base pairs with one or more codons in the mRNA which encode a certain amino acid.

The protein synthesis process is illustrated in Figure 1.14. The ribosome slides along the mRNA reading codon by codon, and facilitates the binding of the tRNA with the corresponding anticodon to the mRNA. The tRNA is carrying the appropriate amino acid, which is removed and attached to the C terminus of the protein chain under construction. The tRNA is then detached and can be recharged and reused. Translation ends when the ribosome reaches a stop codon, which does not have a matching tRNA. Instead, a protein known as *release factor* triggers the release of the completed protein from the ribosome [87].

1.3.3 DNA replication

DNA replication is fundamental for cell division in all living organisms, and constitutes the basis for biological inheritance. During DNA replication, each of the two strands of the original DNA serves as a template for the creation of a new strand. The two resulting DNA molecules will inherit one strand from the original DNA and a newly synthesised complementary strand.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

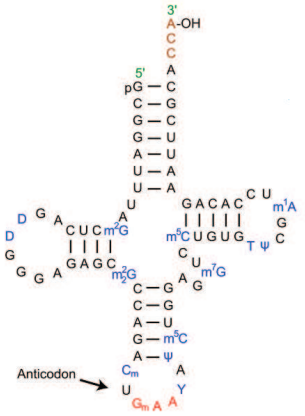

Figure 1.13: Cloverleaf structure of the tRNA.

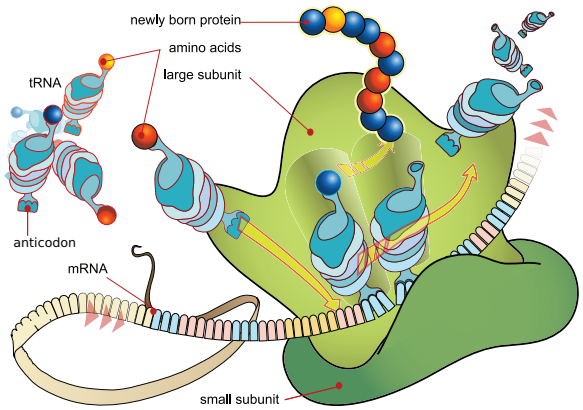

Image source: http://en.wikipedia.org/wiki/File:Ribosome_mRNA_translation_en.svg

Figure 1.14: Protein synthesis.

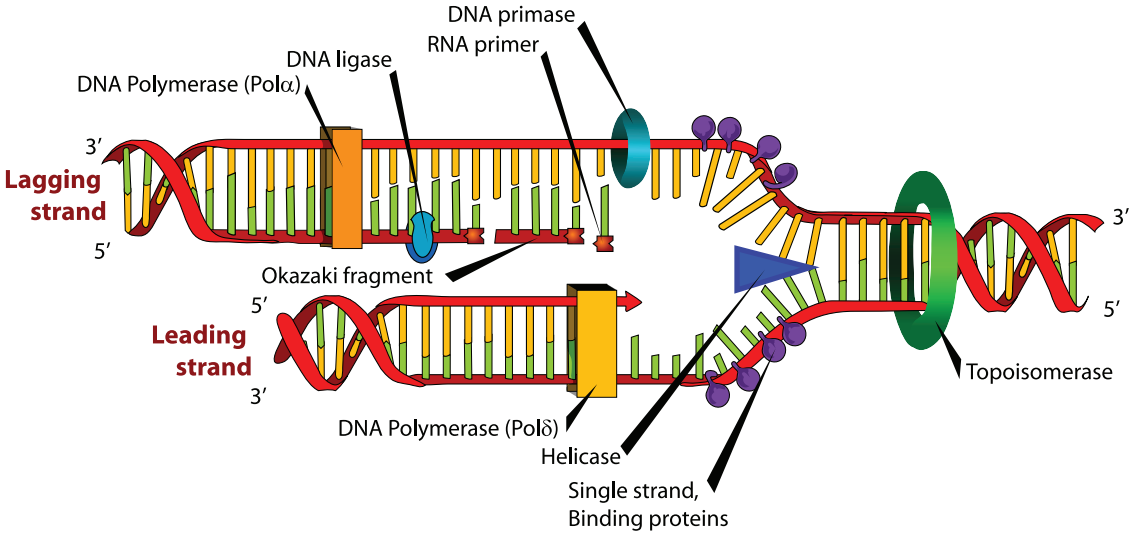

Image source: http://en.wikipedia.org/wiki/File:DNA_replication_en.svg

Figure 1.15: DNA replication. At the "replication fork", the *helicase* breaks the hydrogen bonds in the double helix and *SBB proteins* separate the strands, while the *topoisomerase* keeps the DNA from tangling. On the *leading strand*, the *polymerase* copies the sequence from 5' to 3'. The *lagging strand* is copied also from 5' to 3', in short fragments (*Okazaki fragments*) initiated by *RNA primers* and connected by the *DNA ligase*.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

The replication process is illustrated in Figure 1.15. It is initiated in regions of the DNA molecule called *replication origins* [21] which are rich in adenine-thymine pairs, mainly because these pairs only have two hydrogen bonds, hence being easier to separate [132]. Proteins called DNA helicases and single-strand DNA-binding (SSB) proteins break the double helix and separate the two DNA strands, forming a Y-shaped structure (hence the name "replication fork") [7].

On each strand, a highly accurate self-correcting DNA polymerase enzyme catalyzes nucleotide polymerization in a 5'-to-3' direction, copying the DNA template. The free nucleotides contain 3 phosphate groups, two of which are removed when the nucleotide is incorporared in the chain. The polymerase is not capable of initiating a sequence, as it works by extending the 3' end of an existing sequence paired with the DNA template. To cope with this, short fragments of RNA called *primers* are created by an enzyme called *primase* and serve as starting-points for the polymerase. The two DNA template strands being antiparallel, the 5'-to-3' DNA synthesis can take place continuously on only one of them (the leading strand). On the other (the lagging strand), the synthesis is discontinuous, and occurs in patches of short DNA fragments (approximately 250 base pairs) known as *Okazaki fragments*, initiated by RNA primer molecules [7]. The primers are then replaced with DNA by a polymerase and the fragments are connected by another enzyme called *ligase*, in order to obtain a contiguous strand.

The DNA polymerase has *proofreading* capabilities, being able to remove the nucleotide at the 3' end if that nucleotide does not match the template. The estimated fidelity of DNA polymerization ranges between 10^{-7} and 10^{-8} [114], and is remarkably higher (up to 100 000 times) than RNA polymerization [7]. Yet, although very accurate [141], this error correcting mechanism is not perfect, which places replication errors among the possible causes of changes (mutations) in the DNA sequence: the polymerase may chain the wrong nucleotide and fail to correct it, or may "slip" a few bases (especially in repeating regions, where proofreading is likely to be mislead by the repeated matching pairs) [88].

DNA replication is also achievable *in vitro*, by using DNA polymerases isolated from cells and artificial DNA primers. The technique is called *polymerase chain reaction (PCR)* [148,180], and is based on cyclic enzymatic replications of the DNA. As PCR progresses, the generated DNA becomes a template for replication, maintaining a chain reaction in which the DNA template is exponentially amplified. PCR has various applications in biological and medical research. One of them is DNA cloning involved in several of the sequencing technologies that we discuss in Chapter 2.

1.4 Mutations

Mutations are chemical changes in the DNA sequence of a cell's genome. They are caused either by external factors such as radiation, viruses, mutagenic chemicals [132], by transposons [140] (stretches of DNA can relocate to different positions within the genome of a cell), by internal cellular mechanisms like *somatic hypermutation* (which enables the immune system to adapt and learn to recognize new foreign elements [101]) or by flaws occurring in the process of DNA replication. While some mutations contribute to evolution by enabling the generation of new functions in order to find a proper response to stress conditions [4], many are harmful and are associated with a wide range of medical conditions such as blood disorders, neurological disorders or cancers [24, 41, 84, 98, 116, 132, 143, 149, 161, 196, 217].

Mutations can either affect only several nucleotides, or they can be large-scale modifications of long DNA segments spanning one or more genes that are duplicated, deleted, moved or

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

reversed [88]. We will further give details regarding small-scale mutations, since these are the mutations we mainly aim at revealing with our proposed methods.

Point mutations are substitutions of one nucleotide in the DNA sequence for another type of nucleotide. A substitution of one purine for another (A $\leftrightarrow$ G) or one pyrimidine for another (C $\leftrightarrow$ T) is called a *transition*. Substitutions between purines and pyrimidines are called *transversions*.

Point mutations can have different effects, according to the place where they occur and the nature of the substitution. If the substitution takes place in a non-coding DNA region with no actual function, it will most likely have no effect. Mutations in functional non-coding regions, such as regions that contain signals regulating the initiation and termination of transcription and translation, will have significant effects on the controlled gene's level of expression [88]. Finally, a point mutation in coding DNA may change one amino acid in the sequence translated from that gene, depending on the codon it modifies and the nature of the modification:

- *Synonymous* substitutions change a codon into another that encodes the same amino acid. They are *silent* mutations, i.e. they have no effect on the functioning of the resulting protein. As can be seen in Figure 1.12, most transitions occurring on the third position of a codon are silent. It is also the case of some third position transversions (for amino acids encoded by 3, 4 or 6 codons), as well as for some first position substitutions in the codons of *Arginine* or *Leucine*.

- *Non-synonymous* mutations are substitutions that change the codon so that it no longer encodes the same amino acid. These mutations are classified as *missense* – when the new codon encodes a different amino acid, and *nonsense* – when the new codon is a stop codon.

  Nonsense mutations are almost always harmful since their consequence is the premature termination of the protein. An illustrative example in this case is the nonsense mutation causing cystic fibrosis [24]. Missense mutations may contribute to evolution by helping to create new gene functions, but several have been shown responsible for diseases such as epilepsy [143] or sicke-cell anemia [196] (a disorder causing premature death, but also conferring resistance to the malaria virus), as well as a variety of cancers [41, 98, 149, 161].

A variation in a DNA sequence that consists of a single nucleotide in the genome being different between members of a species, with sufficiently frequent occurrences of the variants within the population (at least 1%), is called a *single nucleotide polymorphism (SNP)*. SNPs are usually considered to be point mutations that have been evolutionarily successful enough to recur in a significant proportion of the population [204].

Insertions and deletions of one or more nucleotides (commonly referred to as *indels*) can also vary in effects depending on where they take place in the DNA sequence. Of particular interest are indels that occur in coding sequences. The triplet nature of gene expression by codons makes them easily corruptible by indels. If the number of inserted or deleted nucleotides is not a multiple of 3 (the length of a codon), the reading frame is disrupted, resulting in a completely different translation after the point of insertion or deletion. Such mutations are known as *frameshift* mutations [194]. They are drastic changes of the coding sequence and are likely to cause loss of function or malfunctioning to the affected gene.

Several known frameshift mutations are associated with diseases such as hereditary hemolytic anemia [84], antithrombin deficiency and thrombosis [160], or renal failure [8].

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

However, the loss of function does not necessarily have completely destructive effects on the organism. As such, some frameshifts are correlated with changes defining an entire species. It is the case of a gene encoding the predominant myosin heavy chain (MYH), inactivated by a frameshift mutation after the lineages leading to humans and chimpanzees diverged [190]. This mutation deprives humans from the powerful masticatory muscles found in other primates, and explains the smaller jaw conformation. Another classic example of frameshift is given by the alleles of the ABO gene associated with the ABO blood group system [50, 211] in humans and several other animals, which basically gives a classification depending on the types of antigens (molecules recognized by the immune system) present in the organism. The ABO locus has three main allelic forms: A, B, and O. The products of A and B have enzymatic activities, and catalyze modifications of the antigen present in all blood types (the H antigen). The O allele differs from the A allele by of a single missing nucleotide, causing frameshift and thus encoding an almost entirely different protein without this enzymatic activity, hence the H antigen remains unmodified in individuals that only possess the O allele.

Moreover, some frameshifts were shown to have triggered functional diversification. Several cases of functional genes created via duplications followed by frameshifts were reported by a study on the human genome [159]. Additionally, in [82], the authors determined some human-specific frameshift mutations producing functional proteins by comparison with the chimpanzee genome. Another experiment consisting of screening an exhaustive set of vertebrate gene families showed that, when a second functional copy of the original gene compensates for this mutation, frameshift mutations can be retained for millions of years and enable new gene functions to be acquired [172]. As a final example, [64] shows that frameshift mutations are involved for structural and functional diversification of the venom system in the advanced snakes. These findings supports the idea that frameshift mutations, although very disruptive, can be an instrument of evolution.

In conclusion, mutations cause genomic sequences to diverge, and their effects – if any – on the affected organism vary from function diversification to function loss or disease. The common origins of genomic sequences of different individuals or species and the mutational events that may have caused their divergence can be inferred via sequence comparison methods. An overview of the most common approaches for biological sequence comparison is given in Chapter 3.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 2

DNA Sequencing

To be able to perform computational analysis of the information enclosed in the genome, the *DNA sequence* (i.e. the exact order of nucleotides in the DNA molecule) needs to be determined. This is achieved via *DNA sequencing* methods. In this chapter, we give an overview of several sequencing methods, from the initial approaches [139, 181] to the platforms most used currently [1, 61, 137, 202] that combine parallel chemical processes and digital imaging to boost the sequencing throughput, and to the newly emerging methods that promise to reduce or eliminate the need for expensive reactants and hardware [25, 200]. Among these technologies, we focus on the SOLiD [1] platform, which features a specific error-correcting read encoding, and for this reason requires read mapping methods that are adapted to the encoding's properties.

2.1 First approaches

Historically, the first notable sequencing methods were proposed by Maxam and Gilbert [139], and by Sanger and Coulson [181] respectively.

2.1.1 Chemical sequencing: the Maxam method

The Maxam method, also known as *chemical sequencing*, relies on radioactive labeling at one end of the DNA molecules and 4 separate reactions of targeted chemical destruction of each of the 4 types of nucleotides, producing labeled fragments, the length of which marks the position of a certain type of nucleotide in the sequence. The obtained fragments are then separated by size via *gel electrophoresis*, a technique consisting of applying an electric field in order to move molecules through a gel. Under the electric field, the molecules will migrate in the gel at different rates, depending on their mass, which enables their separation by size. Consequently, the sequence can be inferred from the relative position in the gel of each labeled fragment [139]. Initially very popular, this approach was then completely replaced by the Sanger method, which had the advantage of relying on fewer dangerous chemicals and less radioactivity.

2.1.2 The chain-terminator method: Sanger

The Sanger approach is inspired by the DNA replication process. It involves the use of a DNA polymerase and DNA primers to produce radioactively or fluorescently labeled DNA fragments of all possible lengths, with the single-stranded DNA molecule to be sequenced as a template. The core idea is the use of nucleotides containing a *dideoxyribose* sugar (Figure 2.1) that cannot link to another nucleotide at the 3' end. Once such a nucleotide is chained by the polymerase

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

to the DNA fragment under construction, the extension is no longer possible and the fragment is ended [181], hence the name of *chain-terminator method*.

Basically, a classic Sanger sequencing experiment begins with the amplification of the template, either by cloning the randomly fragmented DNA into a high-copy-number plasmid implanted in *Escherichiae coli*, or via PCR with primers flanking the target [185]. Templates then enter a "cycle sequencing" reaction [185], consisting of successive steps where the DNA strands first are separated, then the primers, chosen to be complementary to a known sequence immediately flanking the region of interest [185], are attached to the single-stranded template and are subsequently extended. As mentioned above, a certain proportion of the nucleotides used in the extension step are dideoxynucleotides , which force the extension to terminate. Either the primers or the dideoxynucleotides are labeled with a fluorescent dye, the latter option allowing the four reactions for detecting the four types of nucleotides to be carried simultaneously in the same recipient [170]. In consequence, the label on each fragment resulting from this extension will correspond to the nucleotide identity on its terminal position [185]. A well-proportionate combination of deoxynucleotides and dideoxynucleotides will yield extension products with labels for each position of the template.

The actual sequence is obtained by reading the result of electrophoretic separation of the labeled fragments, via laser excitation of the fluorescent dyes and detection of the four color emission intensities (Figure 2.2). The resulting gel image, consisting of bands of four different colors, each band corresponding to the fragments of a certain length (also known as a Sanger sequencing "trace"), is then submitted to computer analysis for translation into a base sequence called a *read* [185].

Sequencing error probabilities can be additionally inferred by the dedicated read translation software, and actual quality values for each read base are computed in order to provide reliable accuracy measures. A first program to obtain such quality scores for Sanger sequencing was Phred [58,59]. The reading error probability calculation relies on parameters obtained from data for which the correct sequence is known, and the quality score Q_i of the base at position i in the inferred sequence depends logarithmically on the error probability P_i for that position, as shown in equation (2.1):

$$Q_i = -10log_{10}P_i. \tag{2.1}$$

Similar quality-scoring methods were later developed for newer sequencing technologies.

Variations of the Sanger method have been intensively used for a wide variety of sequencing tasks for over three decades [185]. Nevertheless, the new biological and medical research challenges call for faster, cheaper methods that facilitate reading and analyzing entire genomes of complex organisms within days, at an affordable cost [183].

2.2 Next generation sequencing

The most important advantage of next generation sequencing technologies is the massively parallel throughput, enabling the processing of millions of sequence reads simultaneously [136], which is a significant improvement over previous technologies. On the down side, the length of the reads currently ranges from 35 to 250 base pairs, whereas the reads produced by advanced implementations of the Sanger method can reach 1000 base pairs [185].

Several methodological choices contribute to the aforementioned throughput leap [136, 185]. Reads are produced from fragment "libraries", obtained by random fragmentation of DNA and

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

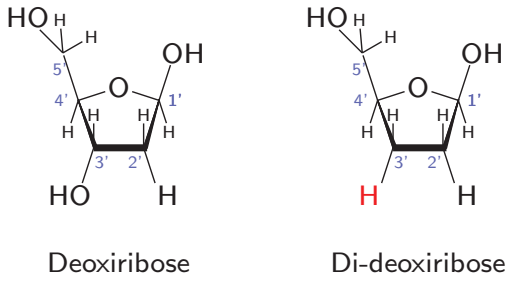

Figure 2.1: The dideoxyribose (right) differs from deoxyribose (left) by the absence of an OH group linked to the 3' carbon. This prevents a nucleotide containing this sugar from bonding to the phosphate group of another nucleotide during DNA polymerization.

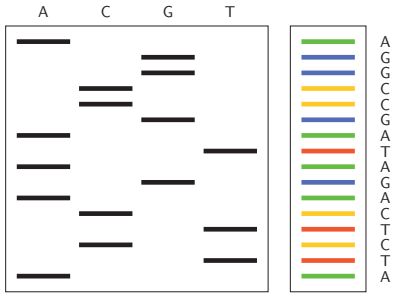

Figure 2.2: Reading the image of electrophoretically separated extension products: the lane (left) or the dye (right) corresponds to the identity of the last nucleotide in the fragments of a certain mass and length, that form a distinct line in the ladder-pattern.

ligation of common adapter sequences, either isolated or mate-paired with controllable distance distributions. The amplification step is designed to reduce biases, hence the *Escherichiae coli* transformation is completely replaced with advanced approaches such as emulsion PCR[1] [53] (Roche/454, SOLiD) or bridge PCR[2] [2, 61] (Illumina/Solexa). In all cases, PCR amplicon derived from a single molecule are spatially clustered, which facilitates the further processing. Generally, the sequencing is performed by synthesis (extension of primers according to a template), using either a polymerase (Roche/454, Illumina/Solexa) or a ligase (SOLiD). The synthesis steps alternate with image acquisitions of the entire array of templates in progress, and the successive captures of light signals carrying information about the molecules incorporated at each step are then interpreted to infer the corresponding nucleotide sequence.

The remainder of this section discusses the particularities of the three main DNA sequencing platforms that are currently available commercially.

2.2.1 Pyrosequencing: Roche/454

The 454 system [137] works on the principle of "pyrosequencing" [176], which engages the pyrophosphate (or diphosphate) molecule released on nucleotide incorporation by the polymerase in a set of reactions having as final result a light emission [136].

The platform uses DNA templates prepared by emulsion PCR. The resulted beads are deposited into individual wells, together with other reactants (polymerase that catalyzes the sequencing by synthesis, sulfurylase and luciferase that facilitate the light reaction). Individual nucleotides are then added in a predetermined order. The incorporation of a nucleotide releases a pyrophospate that produces light in the presence of other reactants, which is captured by a *charged-coupled-device (CCD)* camera. The light signal is generally proportional to the number of incorporated nucleotides if that number is up to approximately 6. This allows the machine to correctly determine the length of small homopolymers. Concerning sequencing errors, the reads obtained with this technique are more likely to be affected by sequencing errors derived from miscounting the incorporated nucleotides, i.e. insertions and deletions, especially in homopolymers, rather than substitutions [144].

[1] DNA molecules and primer-coated beads are isolated in droplets of water in oil.

[2] Primers are covalently linked to a solid-support surface, forcing amplicons to remain immobilized and clustered in a single physical location.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

2.2.2 Cyclic reversible termination: Illumina/Solexa Genome analyzer

The Illumina/Solexa Genome analyzer [61, 202] uses a technique called *cyclic reversible termination*, which basically implies cycles of incorporating a fluorescently modified chain terminator nucleotide that represents the complement of the template base, capturing an image that helps to determine its identity, and finally cleaving off the fluorescent dye and restoring the 3' OH group of the nucleotide, which enables the extension in the next cycle [144].

Within this platform, libraries are composed of adapter-flanked fragments with lengths that reach several hundred base-pairs. The amplification process relies on bridge PCR [61], where both forward and reverse PCR primers are immobilized on a solid substrate, such that all amplicons produced from any single template molecule remain clustered to a single physical location [185]. The resulting single stranded amplicons enter a cyclic sequence interrogation process, as explained above. The most common sequencing errors that occur when using this technique are nucleotide substitutions [144].

2.2.3 Sequencing by ligation: SOLiD

The AB SOLiD technology [43, 165] (**S**equencing by **O**ligo **Li**gation and **D**etection), based on the work described in [186] and [142], uses dye-labeled oligonucleotide probes in a sequencing-by-ligation process catalyzed by a DNA ligase, one probe determining *two nucleotide positions from the template at a time*. The sixteen possible 2-base combinations are encoded by four fluorescent dyes (Figure 2.3 **e**), and the color-dinucleotide association features some error correcting properties [1] that we will discuss below.

Basically, this sequencing technique follows several steps. First, the genomic DNA to be sequenced is fragmented, and two specific adapter sequences are attached to the 5' and 3' ends of each fragment. The fragments are amplified by emulsion PCR [53] on the surface of magnetic beads, eventually resulting in beads enriched with PCR products from a single template, which are deposited onto a glass slide.

Further, the actual sequencing process begins by annealing a universal sequencing primer that is complementary to the adapters on the library fragments (Figure 2.3 **a**). Subsequently, fluorescently labeled 8-mer oligonucleotides are successively hybridized on the template. The oligonucleotides contain 6 universal bases (that can form pairs with any base on the template) and two adjacent nucleotides at the 3' end that identify a pair of bases at the corresponding position on the template, correlated with the identity of the fluorescent labels at their 5' end. After annealing and ligation of the oligonucleotide at the 5' end of the primer, the fluorescent signal of the label is captured and measured, and bases 6-8 are then cleaved off, along with the fluorescent dye, leaving 5 nucleotides to extend the primer, with a free 5' phosphate group available for the next ligation (Figure 2.3 **b**). Hence, positions $p+1$ and $p+2$ are correctly base-paired after attaching one oligonucleotide, followed by the bases at distance 5 ($p+6$ and $p+7$) being correctly paired with the next oligonucleotide, as shown in Figure 2.3 **c**. The nucleotides at positions that do not fit this pattern are determined in subsequent rounds. Five rounds consisting of several ligation cycles are necessary to cover the template (Figure 2.3 **f**). At the beginning of each round, the position of the primer is shifted by one, thus enabling the reading of a different subset of bases (Figure 2.3 **d**).

The colored labels registered after each ligation describe all the pairs of adjacent nucleotides in the template. A SOLiD read (Figure 2.3 **g**) is a sequence of colors, each color representing two nucleotides in which the second base of each dinucleotide unit constitutes the first base of the following dinucleotide. The color sequence is prefixed by a base symbol (the last nucleotide of

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

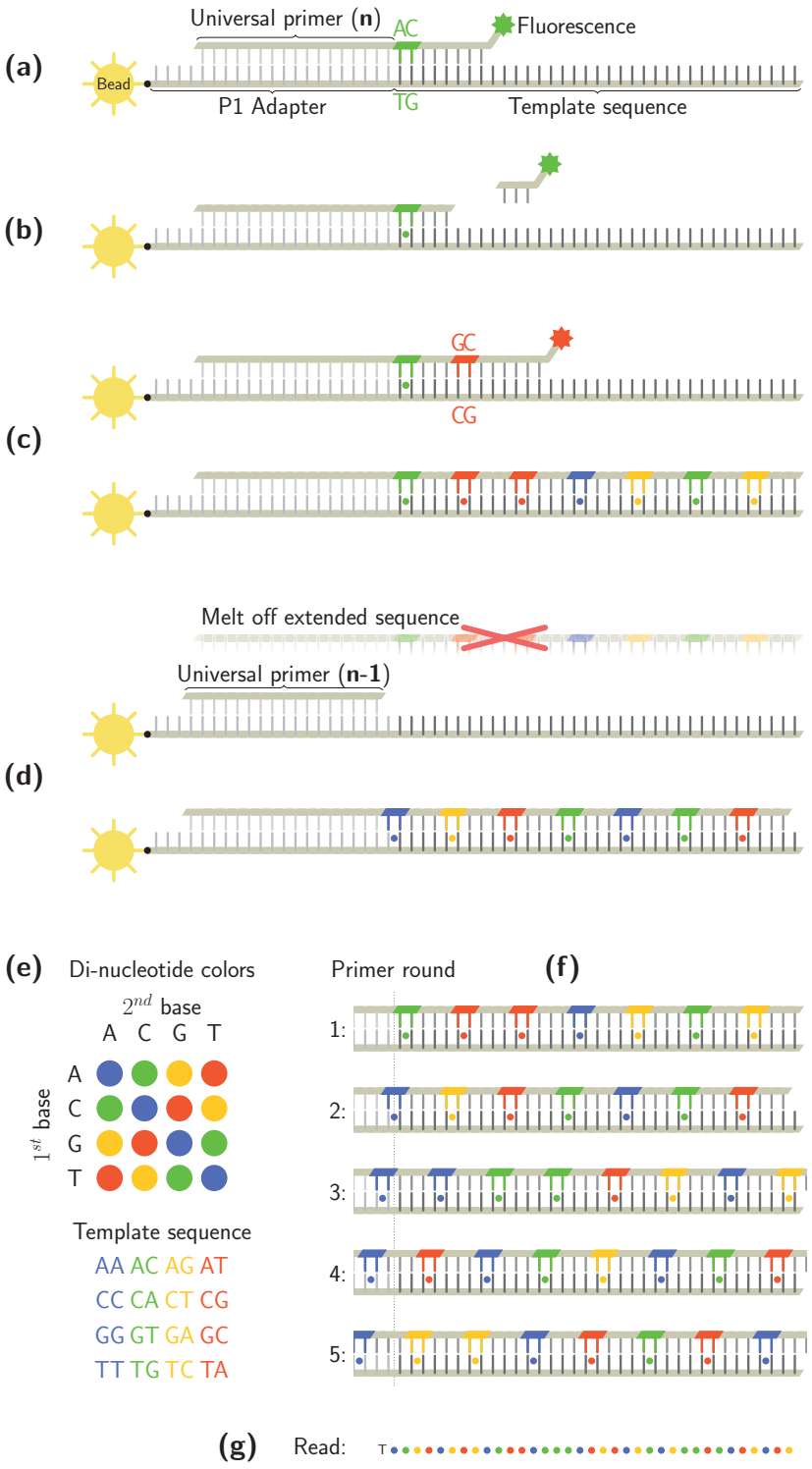

Figure 2.3: Sequencing by ligation technique used in the SOLiD technology. **(a)** Primer annealing and hybridization of a fluorescently labeled 8-mer oligonucleotides on the template. **(b)** The fluorescent signal of the label is captured and measured, and bases 6-8 are then cleaved off. **(c)** Positions $p+1$ and $p+2$ are correctly base-paired after attaching one oligonucleotide, followed by the bases at distance 5 ($p+6$ and $p+7$) being correctly paired with the next oligonucleotide. **(d)** At the beginning of each ligation round, the position of the primer is shifted by one. **(e)** Di-nucleotie colors table. **(f)** Five rounds consisting of several ligation cycles are necessary to cover the template. **(g)** Example of a SOLiD read.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

the primer) which helps the translation without ambiguities. The SOLiD encoding is explained in detail later in Section 8.1. A quality prediction algorithm obtains Q values that conform to the industry-standard relation [66, 203] established by the Phred software [58].

2.3 Next-next generation sequencing

The next steps in DNA sequencing aim at reducing or eliminating the need for expensive components such as fluorescent molecular labels and optical hardware for base differentiation, as well as fragment library amplification which, besides from being costly and time-consuming, is also a possible source of errors and data inaccuracies.

2.3.1 Semiconductor sequencing: Ion Torrent

Ion Torrent [200] implements a technique called *semiconductor sequencing*, which combines semiconductor technology with a simple sequencing chemistry and creates a direct connection between the chemical information and the digital information. As in many other platforms, the sequencing is catalyzed by a polymerase which copies a DNA template. This biochemical process takes place in a massively parallel way, in a high-density array of micro-machined wells, each well containing a different DNA template. A certain type of nucleotide (A, C, T or G) is made available at each cycle. If that nucleotide is incorporated by the polymerase, a hydrogen ion is released as a byproduct, and its charge will change the pH of the solution. This modification is detected by the ion sensor, and the base is immediately taken into account [169]. If several bases of the same type are incorporated in the same cycle, a hydrogen ion is released for each of them and a higher voltage is generated, allowing the sensor to account for the right number of consecutive identical bases. If the base from the current cycle does not match, the pH remains unchanged and the sensor is not activated.

2.3.2 Nanopore sequencing

Another innovative technology [191, 199] consists of detecting molecules as they are driven through nanopores. The system relies on nanopores [52] bounded by an engineered protein called *alpha-hemolysin*, introduced in a lipid bilayer. A pair of electrodes are placed on either side of the bilayer, which has a high electrical resistance, thus forcing the current to flow only through the nanopore. The nanopores are coupled with an enzyme called *exonuclease* that is responsible for capturing DNA strands, sequentially cleaving individual bases from the strand, and directing the bases into the aperture of the nanopore. As individual DNA bases sequentially pass through the nanopore, they temporarily bind to an engineered *cyclodextrin*[3] *sensor*, placed inside the surface of the nanopore, disturbing the current and creating a characteristic signal for each type of base. Consequently, the electrical current trace provides a record of the sequence of bases passing through the nanopore. The technology is still being perfected, and more accurate ways of determining the base identity are being explored [192].

2.4 Applications

The advances in DNA sequencing, and new generation sequencing in particular, have given a boost to a wide range of genomics research fields, improving the technical and financial feasibility

[3] Compound made of sugar molecules bound together in a ring.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

of a wide range of sequencing experiments [205], with applications in evolutionary biology, medicine and drug design. Recently, these techniques have also made possible genome-scale analyses of genomes of extinct organisms [77, 78].

The sequencing process generates a large collection of *reads*, i.e. sequences that correspond to fragments of the nucleic acid molecule(s) given as input. Depending on the data source and intended application, these reads are either mapped to a reference sequence or assembled *de novo*. Generally, the technologies producing longer reads (such as Roche/454) are better suited for assembling genomes in the absence of a reference, while resequencing tasks can be successfully accomplished with both long- and short-read technologies [205]. Actual applications include genome analysis, metagenomics, targeted sequencing or transciptome sequencing.

The high-throughput of new generation sequencing technologies facilitates the *sequencing of entire genomes*, from microbes to humans, with multiple scientific benefits [205]. For instance, in disease-causing bacteria and viruses, which tend to evolve continuously in order to gain resistance to antibiotics and antivirals [136], the identification of mutations in a certain resistant strain is achieved by complete genome sequencing and comparison (read mapping) to a reference (usually non-virulent) genome [89, 206]. Also, the recent sequencing of the genome of cytogenetically normal acute myeloid leukemia cells allowed the identification of novel, tumor-specific gene mutations [121].

Cancer research also benefits from *targeted genomic resequencing*, i.e. sequencing genomic subregions and gene sets in order to detect polymorphisms and mutations in genes thought to be implicated in cancer.

On a larger scale, new generation sequencing technologies have a key role in *metagenomics*, the study of microbial diversity in environmental and clinical samples. Basically, genomic DNA extracted from such samples is sequenced, and the result is either aligned to known reference sequences for microorganisms to support the presence of species related to them in the analyzed sample, or submitted to *de novo* assembly to infer the presence of known and potentially new species. In addition, analysis of the number of reads found to correspond to each species provides quantitative information regarding the sample's composition [205].

Transciptome sequencing is involved in mapping and quantifying transcripts in biological samples. In this context, gene expression levels can be deduced from the number of reads that map to an exon of a gene, and both qualitative and quantitative information regarding splicing diversity can be inferred [205].

Regardless of the protocol and application, any sequencing task depends on subsequent computational processing of the reads, either for assembly or for mapping on a reference genome. Part III of this thesis discusses read mapping software, focusing on methods designed for the SOLiD technology.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 3

Algorithms and tools for biological sequence alignment

Sequence comparison is a fundamental instrument for understanding the origins and functions of biological sequences. In living organisms, genetic diversity is achieved by alterations of pre-existing sequences rather than *de novo* creation. While DNA sequences with no genetic function can diverge without restriction, for coding and regulatory regions the diversification is balanced by a *purifying selection* mechanism, which tends to eliminate individuals carrying mutations that disrupt important genetic functions. Consequently, genes with similar functions usually share important similarities at the nucleotide sequence level [4]. Although sequence similarity does not guarantee relatedness and unmistakable function homology, it is considered strong evidence in this direction and constitutes the basis for analyzing and predicting the meaning of newly discovered sequences [54].

In this chapter we discuss state-of-the-art methods and tools for similarity search. After a short reminder of the representations used for biological sequences, we introduce the notion of sequence alignment and present the classic *dynamic programming algorithms for optimally aligning two sequences*, as well as the most common similarity measures employed for comparing DNA and proteins respectively. We continue with an overview of *heuristic methods*, which are less accurate, but faster and therefore better suited for comparisons of large databases. Finally, among the different heuristics we focus on the technique of *spaced seeds*, and discuss their properties and approaches for their design.

3.1 Biological sequence representation

From the computational point of view, the biopolymers discussed in Chapter 1 are strings over an alphabet of symbols representing their respective structural units. As such, in our context, a DNA sequence has the representation of a string over the four-letter alphabet $\mathcal{N} = \{A, C, G, T\}$ (standing for the four bases *adenine*, *cytosine*, *guanine* and *thymine*). Note that, since the two DNA strands are complementary, it suffices to store the sequence of one strand only. Similarly, the RNA sequence is a string over the four-letter alphabet $\mathcal{N}' = \{A, C, G, U\}$ (where U is the symbol of *uracil*), and proteins sequences are usually strings over the 20-letter alphabet of amino acid symbols: $\mathcal{A} = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$. To handle ambiguities at a given position in a nucleotide sequence, the IUPAC nucleotide codes [42], given in Table 3.1, provide symbols for all possible subsets of bases. Several groups of very similar amino acids can also be denoted by a single symbol, as can be seen in Table 3.2.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Code	Base
A	Adenine
C	Cytosine
G	Guanine
T	Thymine
U	Uracil

Code	Base
R	A or G (Purines)
Y	C or T (Pyrimidines)
S	G or C
W	A or T
K	G or T
M	A or C

Code	Base
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base

Table 3.1: IUPAC nucleotide codes.

Code	Amino Acid
A (Ala)	Alanine
C (Cys)	Cysteine
D (Asp)	Aspartic Acid
E (Glu)	Glutamic Acid
F (Phe)	Phenylalanine
G (Gly)	Glycine
H (His)	Histidine
I (Ile)	Isoleucine
K (Lys)	Lysine
L (Leu)	Leucine

Code	Amino Acid
M (Met)	Methionine
N (Asn)	Asparagine
P (Pro)	Proline
Q (Gln)	Glutamine
R (Arg)	Arginine
S (Ser)	Serine
T (Thr)	Threonine
V (Val)	Valine
W (Trp)	Tryptophan
Y (Tyr)	Tyrosine

Code	Amino Acids
B (Asx)	Aspartic acid or Asparagine
J (Xle)	Leucine or Isoleucine
Z (Glx)	Glutamine or Glutamic acid
X (Xaa)	Any amino acid

Table 3.2: Amino acid codes.

3.2 Pairwise sequence alignment

Since biological sequences are represented as strings over a certain alphabet, their comparison methods are strongly related to string editing. In Section 1.4 we briefly presented various types of mutations that can affect genetic sequences and cause their divergence, in particular point mutations and indels. From a string editing perspective, these correspond to operations of character substitution, or insertion/deletion of one or several characters in a string respectively.

The alignment of two or more sequences is a procedure which allows one to identify series of similar (conserved) characters that appear in the same order in the aligned sequences, and eventually to infer a set of modifications (substitutions and indels) involved in the transformation of one sequence into another. In the following, we will not give details regarding methods of simultaneously comparing multiple sequences, but rather focus on pairwise sequence alignment. The next sections present basic, state-of-the art sequence alignment algorithms. Nonetheless, the literature provides a plethora of variations and adaptions of these algorithms to specific problems, such as aligning coding DNA [16, 79, 85, 106, 166, 168, 214] by making use of its triplet nature imposed by codons, or constraining functional sites (motifs) to be aligned together in protein alignment [15, 39, 113].

3.2.1 Definition and representation

Definition 1. *Given two sequences $S = s_1...s_n$ and $T = t_1...t_m$ over an alphabet Σ (where Σ is either the alphabet of nucleotides for nucleic acid sequences, or the alphabet of amino acids for proteins) their alignment consists of two equally sized sequences $S' = s'_1...s'_l$ and $T' = t'_1...t'_l$, obtained by inserting zero or more gaps (represented by a symbol $- \notin \Sigma$) between the symbols of S and T respectively, in order to shift on the same position intervals subsequences that are*

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010


```
ATATTT--GATATT-CAACTCGACTGGTGTA---ATTCGAG
||.|||  |||||| |:||||   |||||||   ||.|:||
ATTTTTACGATATTTCGACTC---TGGTGTACTCATGCAAG
```

(a)

```
GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL
GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL
```

(b)

Figure 3.1: Examples of pairwise alignment between two DNA sequences (a) and two proteins (b).

similar in both S and T, with the constraint that $\nexists h \in \{1, ..., l\} : s'_h = t'_h = -$.

The two symbols at position i in S' and T', s'_i and t'_i, are aligned. If both s'_i and $t'_i \in \Sigma$, then this aligned pair corresponds to either a *match* when $s'_i = t'_i$, or a *mismatch* (*substitution*) when they are different. If one of the symbols is a gap, then this aligned pair marks an *indel* [212]. By convention, gaps cannot occur at the same position in both S' and T'.

The alignment of two sequences S and T is typically represented as a matrix, where each sequence occupies a row with a character per column, with gaps denoted by a symbol "-"$\notin \Sigma$ that may be inserted between the characters (Figure 3.1). An additional row, usually placed between those corresponding to sequences, gives a "summary" of the similarities encountered within this alignment. For DNA sequences (Figure 3.1 a), the symbol "|" conventionally denotes matches, and "." stands for mismatches. Some alignment tools [157] distinguish between transversion mismatches (purine-pyrimidine pairs) and transition mismatches (when the aligned nucleotides are both purines or both pyrimidines) by marking the latter with a different symbol ":" in the alignment representation. For protein alignment (Figure 3.1 b), pairs of identical characters are marked by their corresponding symbol, and other related pairs are simply denoted by a "+" [102].

3.2.2 Score

The alignment provides information regarding mutations that may have been involved in the divergence of the two sequences, under the hypothesis that the sequences have a common ancestor. The relatedness of the aligned sequences is quantified by the *score* assigned to their alignment, which comprises the scores associated to individual pairs of aligned characters, reflecting their similarity. Typically, identical or related character pairs are rewarded with a positive score, while gaps and pairs of characters that do not match are penalized with a negative value.

Substitution scores The scores basically aim at expressing the relative likelihood that sequences are related as opposed to being unrelated. For example, DNA sequence alignments are often scored under a simple $+m$ (for matches) / $-x$ (for mismatches) system, where (m, x) is a pair of integer values, for example $(5, 4)$. On the other hand, amino acids similarities are more complex and cannot be properly expressed in simple terms of match and mismatch. Generally, mutations that result in amino acids with the same physical and chemical properties (size, charge, polarity, presented in Figure 1.8) as the original are more likely to survive natural

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

selection, since the resulting protein is likely to be capable of carrying the same function. Consequently, some substitutions are more likely to be encountered in evolving sequences than others, and this fact needs to be reflected in the score for a proper evaluation of the sequence similarity.

Formally, a score $\sigma(a, b)$ (where a, b denote either nucleotides or amino acids) can be obtained [55, 104] from

- the joint probability q_{ab} that a and b have been derived from a common ancestor, and
- the probability that a and b occur by chance with their respective frequencies p_a and p_b.

as the *log-odds ratio* of these probabilities:

$$\sigma(a, b) = \log\left(\frac{q_{ab}}{p_a p_b}\right). \tag{3.1}$$

Indeed, the obtained values $\sigma(a, b)$ will be positive for (a, b) pairs that are more likely to have a common ancestor than to occur by chance, and negative in the opposite case.

The overall score $\hat{\sigma}_{ungapped}(S_1, S_2)$ of the ungapped alignment of two sequences S_1 and S_2 of equal length is given by the log-odds ratio of the two probabilities of divergence from a common ancestor and independent occurrence respectively, under the assumption that mutations at different positions in the sequences have occurred independently:

$$\hat{\sigma}_{ungapped}(S, T) = \log \frac{\prod_i q_{s_i t_i}}{\prod_i p_{s_i} \prod_i p_{t_i}} = \log \prod_i \frac{q_{s_i t_i}}{p_{s_i} p_{t_i}} = \sum_i \log \frac{q_{s_i t_i}}{p_{s_i} p_{t_i}} = \sum_i \sigma(s_i, t_i) \tag{3.2}$$

where s_i and t_i are the symbols occurring at position i in S and T respectively, hence the additive nature of sequence alignment scores.

Section 3.3 presents several widely used scoring schemes for evaluating pairs of nucleotides or amino acids, generally obtained via empirical studies on manually curated ungapped alignments.

Gap penalties In addition to matching and substitution scores, costs (denoted $\gamma(g)$) are associated with gaps depending on their length g. In practice, for efficient implementations of alignment algorithms (discussed in Section 3.2.3) the dependence is typically either linear

$$\gamma(g) = -g \cdot d \tag{3.3}$$

or affine

$$\gamma(g) = -d - (g - 1) \cdot e \tag{3.4}$$

where d is the penalty for gap opening, and e is the penalty for gap extensions. Gap extensions are usually less penalized than gap openings, which makes affine gap costs useful whenever gaps spanning several positions are expected almost as frequently as gaps of size one [55].

Let $S' = s'_1 ... s'_n$ and $T' = t'_1 ... t'_n$ be two sequences obtained by inserting gaps in S and T in order to obtain their gapped alignment. The score $\hat{\sigma}_{gapped}(S, T)$ of the gapped alignment of two sequences S and T is given by the summed scores of the aligned symbols, added to the penalties corresponding to each gap, namely:

$$\hat{\sigma}_{gapped}(S, T) = \hat{\sigma}(S', T') = \sum_{s'_i \in \Sigma, t'_i \in \Sigma} \sigma(s'_i, t'_i) + \sum_{\substack{[s'_i \ldots s'_{i+g-1}] = \{-\}^g \\ \text{maximal gap}}} \gamma(g) + \sum_{\substack{[t'_i \ldots t'_{i+g-1}] = \{-\}^g \\ \text{maximal gap}}} \gamma(g). \tag{3.5}$$

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

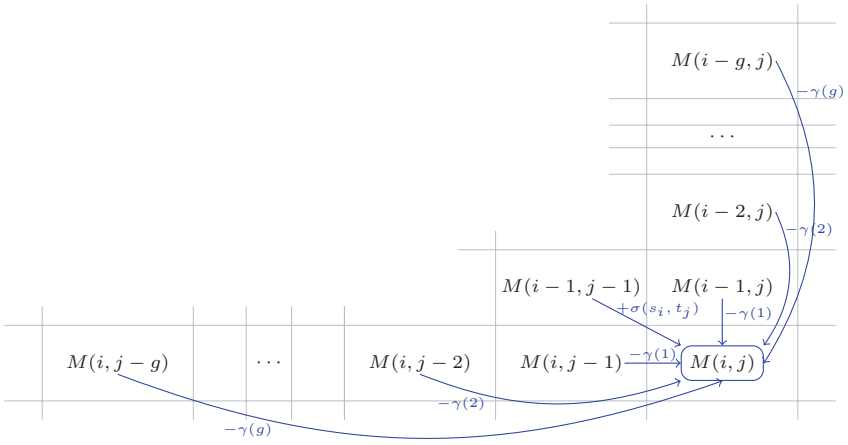

Figure 3.2: General dynamic programming alignment algorithm: illustration of the recurrence relation.

3.2.3 Pairwise alignment algorithms

The optimal gapped alignment between two sequences $S = s_1...s_m$ and $T = t_1...t_n$ is typically obtained via dynamic programming approaches that aim at maximizing the alignment score. Depending on the type of problem that requires sequence comparison, one may be interested either in aligning the sequences *globally*, or in finding *local* similarities, i.e. similar subsequences. However, regardless of whether the sought similarities are global or local, the algorithms basically follow the same protocol: the optimal alignment is recursively built using previous solutions for optimal alignments of smaller substrings.

Like in all dynamic programming approaches, these partial solutions are stored in a two dimensional table M for re-usage at subsequent steps: $M(i,j)$ stores the score of the best alignment obtained between the subsequences $s_1...s_i$ and $t_1...t_j$. The basic idea is illustrated by Figure 3.2 and formalized in equation (3.6). The recurrence corresponds to the computation of global alignments. The detection of local similarities requires a modification in order to discard partial solutions with negative scores, which we will discuss below in the context of local alignment algorithms. Knowing all the best solutions for aligning the subsequences $S_{1..k} = s_1...s_k$ and $T_{1..h} = t_1...t_h$ of S and T for all $k < i$ and $h \leq j$ or $k \leq i$ and $h < j$, the best alignment of $S_{1..i} = s_1...s_i$ and $T_{1..j} = t_1...t_j$ is obtained by extending either the alignment of $S_{1..i-1}$ and $T_{1..j-1}$ with the pair formed with the next symbols in each sequence (s_i, t_j) (a), or by adding a gap of length g to the alignment of one of the sequences with a prefix of the other (b and c).

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + \sigma(s_i,t_j) & & (a) \\ M(i-g,j) - \gamma(g), & \forall g \in \{1,...,i-1\} & (b) \\ M(i,j-g) - \gamma(g), & \forall g \in \{1,...,i-1\} & (c) \end{cases} \tag{3.6}$$

The original algorithms for aligning globally [150] and locally [188] two sequences of lengths m and n were designed for arbitrary gap penalty functions and had an execution time of $\Theta(m^2 n)$ [75]. The modification proposed later in [75] reduces the complexity to $\Theta(mn)$ by establishing the affine dependence of the gap penalty costs, and thereby reducing the depth of the partial solution dependence and the number of partial solutions taken into account at each step by allowing growing gaps to be evaluated progressively. Instead of a single table for storing partial solutions, this approach uses 3 tables: M storing all partial solutions consisting of alignments that end with matches, and G_S, G_T storing the scores of alignments that end with

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

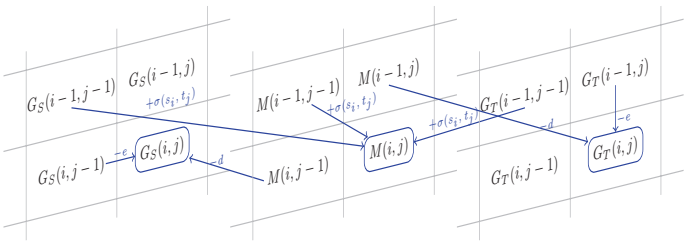

Figure 3.3: Sequence alignment with affine gap costs: illustration of the recurrence relation.

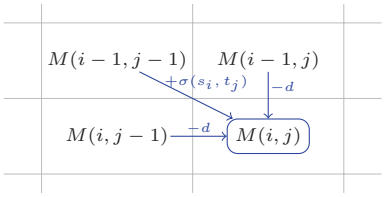

Figure 3.4: Sequence alignment with linear gap costs: illustration of the recurrence relation.

gaps in S and T respectively. In this setup, gaps are opened and extended one position at a time, allowing each partial solution to depend only on a small, constant number of previously computed values (equation (3.7), Figure 3.3).

$$\begin{aligned} M(i,j) &= \max \begin{cases} M(i-1,j-1)+\sigma(s_i,t_j) \\ G_S(i-1,j-1)+\sigma(s_i,t_j) \\ G_T(i-1,j-1)+\sigma(s_i,t_j) \end{cases} \\ G_S(i,j) &= \max \begin{cases} M(i,j-1)-d \\ G_S(i,j-1)-e \end{cases} \\ G_T(i,j) &= \max \begin{cases} M(i-1,j)-d \\ G_T(i-1,j)-e \end{cases} \end{aligned} \tag{3.7}$$

For linearly penalized gaps, the dependency formula (equation (3.8), Figure 3.4) becomes less complex, requiring a single table for storing partial solutions and only 3 partial solutions to be evaluated at each step. For simplicity, the following descriptions of local and global alignment algorithms will rely on this linear gap approach.

$$M(i,j) = \max \begin{cases} M(i-1,j-1)+\sigma(s_i,t_j) \\ M(i,j-1)-d \\ M(i-1,j)-d \end{cases} \tag{3.8}$$

Global alignment

The recurrence relation for computing a global alignment between two sequences $S = s_1...s_m$ and $T = t_1...t_n$ with linearly penalized gaps is given by equation (3.8). The algorithm populates a two dimensional table M of size $(m+1) \times (n+1)$. The border cells $M(i,0)$ and $M(0,j)$ correspond to the insertions given by equation (3.9). These partial alignments correspond to

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

the insertion of gaps before one of the sequences.

$$\begin{array}{lcl} M(0,0) & = & 0 \\ M(i,0) & = & M(i-1,0)-d \quad = \quad -id, \quad \forall i \in \{1,...,m\} \\ M(0,j) & = & M(0,j-1)-d \quad = \quad -jd, \quad \forall j \in \{1,...,n\} \end{array} \tag{3.9}$$

The global alignment score is retrieved from $M(m,n)$. The actual alignment can be constructed by *tracing back* from $M(m,n)$ the partial solutions $M(i,j)$ that were chosen at each recursive step, until $M(0,0)$ is reached. This can be done by re-calculating (3.8) for each $M(i,j)$ on the path from $M(m,n)$ to $M(0,0)$ and following the cell providing the maximum score. Alternatively, the coordinates of the cell contributing to each partial solution can be determined at the recursion step and memorized in a separate bi-dimensional table. Let $S' = s'_1...s'_l$ and $T' = t'_1...t'_l$ be strings of equal length l obtained from S and T by inserting gaps in order to construct an alignment with the best score, given by $M(m,n)$. For any $M(i,j)$, let (i_{prev}, j_{prev}) be the coordinates of the cell in M contributing to the partial solution $M(i,j)$. The strings S' and T' are obtained backwards as shown in Algorithm 1.

Local alignment

The local alignment algorithm of two sequences $S = s_1...s_m$ and $T = t_1...t_n$ allows determining similarities between subsequences rather than forcing a global comparison. A local alignment is allowed to start and end at any point in the sequences, and its score must not be influenced by dissimilar regions flanking the aligned ones. In consequence, the algorithm ignores partial alignments between very dissimilar subsequences. To do this, the top-left border of the matrix M is initialized to 0

$$\begin{array}{lcl} M(0,0) & = & 0 \\ M(i,0) & = & 0, \quad \forall i \in \{1,...,m\} \\ M(0,j) & = & 0, \quad \forall j \in \{1,...,n\} \end{array} \tag{3.10}$$

and the recursion step differs from (3.8) by an additional option, allowing the algorithm to discard negative scoring alignments and mark each zero-valued cell as a potential start point of a local similarity:

$$M(i,j) = \max \begin{cases} 0 \\ M(i-1,j-1) + \sigma(s_i, t_j) \\ M(i,j-1) - d \\ M(i-1,j) - d \end{cases} \tag{3.11}$$

The score of the best local alignment is given by the highest value $M(i,j)$ in the matrix. The corresponding local alignment can be retrieved by a *traceback* procedure similar to the one described in Algorithm 1, starting in the best scoring cell and ending at the first cell (i',j') with $M(i',j') = 0$.

Semi-global alignment

The semi-global alignment method allows determining if one of the sequences is *contained* in the other, or if the two overlap on the extremities. This approach is generally useful for mapping or assembly. Basically, the method combines the local alignment initialization given by (3.10) and the global alignment recursion given by (3.8) in order to obtain the best scoring overlaps. The score of the best semi-global alignment is the highest value that appears on the bottom-right border of M ($\max_{i\in\{1,...,m\}, j\in\{1,...,n\}}\{M(i,n), M(m,j)\}$). The alignment can be obtained via *traceback* from this highest scoring cell until the top-left border of M is reached.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Input:

Two sequences S and T;

The dynamic programming table M obtained by the alignment algorithm given in (3.9) and (3.6), holding the best-scoring alignment of S and T.

Output: The best scoring alignment in the form of two strings S' and T'.

begin

$i \leftarrow m$, $j \leftarrow n$, $h \leftarrow 1$

while $i \neq 0$ *and* $j \neq 0$ **do**

$$(i_{prev}, j_{prev}) \leftarrow arg\max \begin{cases} M(i-1, j-1) + \sigma(s_i, t_j) \\ M(i, j-1) - d \\ M(i-1, j) - d; \end{cases}$$

if $i_{prev} == i$ **then**

$s'_h \leftarrow$ ' $-$ ';

else

$s'_h \leftarrow s_i$;

end

if $j_{prev} == j$ **then**

$t'_h \leftarrow$ ' $-$ ';

else

$t'_h \leftarrow t_j$;

end

$h \leftarrow h + 1$;

$i \leftarrow i_{prev}$, $j \leftarrow j_{prev}$;

end

reverse(S');

reverse(T');

return $\{S', T'\}$;

end

Algorithm 1: Traceback algorithm on a dynamic programming matrix M for retrieving the best global alignment between two sequences $S = s_1...s_m$ and $T = t_1...t_n$, of lengths m and n respectively. The score of the best alignment is given by $M(m, n)$. The alignment is obtained in the form of two strings $S' = s'_1...s'_l$ and $T' = t'_1...t'_l$ of equal length l, obtained from S and T by inserting gaps, with $(s'_h, t'_h), h \in \{1, ..., l\}$ giving the aligned symbols.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

3.3 Defining sequence similarity via substitution scores

This section presents how biological sequence similarity scores are obtained in practice, following the methodology described in Section 3.2.2.

3.3.1 Amino acids

PAM matrices

The PAM (Accepted Point Mutation) [51] matrices are among the first amino acid substitution matrices for protein comparison used in practice. *Accepted point mutations* are replacements of one amino acid by another which survive by natural selection. Several PAM matrices for different evolutionary distances between the compared sequences were computed. As a reference, one PAM denotes the evolutionary time period over which 1% of the amino acids in a sequence are expected to be modified by accepted substitutions.

The similarity scores are obtained from mutation patterns observed in alignments of closely related proteins. According to [51], the data was acquired from phylogenetic trees corresponding to 71 families of strongly similar proteins (at least 85% identity between any two within the family), and mutations are considered relatively to the inferred ancestral sequences.

To build the matrix, the numbers of accepted point mutations A_{ab} were first counted for all pairs of a and b occurring at the same position in sequences and their immediate ancestor in the tree. These values are considered to be symmetric, i.e. any encountered pair (a, b) contributes to both A_{ab} and A_{ba}. The probability of substituting a by b is then derived from these counts. Although the original paper provides a more complex reasoning and formula for these probabilities, the calculation can be simplified [35, 55] as

$$q_{ab} = \frac{A_{ab}}{\sum_c A_{ac}}. \tag{3.12}$$

To ensure the 1% expected substitution rate, these probabilities are scaled by a fixed constant λ as follows:

$$M_{ab} = \lambda \frac{A_{ab}}{\sum_c A_{ac}} \tag{3.13}$$

and

$$M_{aa} = 1 - \sum_{c \neq a} M_{ac} \tag{3.14}$$

with λ chosen so that only 1 in 100 amino acids change, i.e.

$$\sum_a p_a \sum_{b \neq a} M_{ab} = \frac{1}{100} \Rightarrow \sum_a p_a \sum_{b \neq a} \lambda \frac{A_{ab}}{\sum_c A_{ac}} = \frac{1}{100} \tag{3.15}$$

with p_a the occurrence probability of amino acid a.

The probability matrix M defined in equations (3.13) and (3.14) can serve as a basis for obtaining mutation probabilities $M^{(n)}$ corresponding to higher evolutionary distances (n PAMs), which are derived from M by matrix multiplication, i.e. $M^{(n)} = M^n$.

The substitution scores result from the mutation probabilities as the *log-odds ratio* given by equation (3.1). In this case, the joint probability that a and b have been derived from a common

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

ancestor is given by the probability of encountering a (denoted p_a) and, given a the probability of substituting a for b, i.e. M_{ab}. Consequently, the score assigned to a pair (a, b) is

$$\text{PAM}n(a,b) = C \log\left(\frac{p_a M_{ab}^{(n)}}{p_a p_b}\right) = C \log\left(\frac{M_{ab}^{(n)}}{p_b}\right) \tag{3.16}$$

where C is a scaling constant originally set to 10, for more precision when truncating scores to integer values.

BLOSUM matrices

With the goal of achieving better precision when aligning protein sequences that are very distant evolutionarily, the BLOSUM (BLOck SUbstitution Matrix) matrices [86] are constructed using multiple alignments of evolutionarily divergent proteins.

The probabilities used in the matrix calculation are computed from multiple ungapped local alignments represented as *blocks* with each row a different protein segment and each column an aligned residue position. More than 2000 blocks of aligned sequence segments were involved in this analysis, characterizing over 500 groups of related proteins. For each block in the database, all possible pairs in each column of each block are counted and summed. A block consisting of s segments of w amino acids contributes with $ws(s-1)/2$ amino acid pairs to the count.

Consider that the amino acids are uniquely labeled by numbers from 1 to 20, and let f_{ij} denote the number of times the pair (i, j) is observed in the data. The probability of occurrence for each (i, j) pair is obtained as

$$q_{ij} = \frac{f_{ij}}{\sum_i \sum_{k \leq i} f_{ik}}. \tag{3.17}$$

The frequency of i is further calculated as

$$p_i = q_{ii} + \frac{\sum_{j \neq i} q_{ij}}{2} = \sum_{j \leq i} q_{ij} \tag{3.18}$$

where the division by 2 is a compensation for the symmetric counts.

Further, for each (i, j) pair, the expected probability e_{ij} of alignment by chance is given by

$$e_{ij} = \begin{cases} p_i p_j & \text{if } i = j \\ p_i p_j + p_j p_i = 2 p_i p_j & \text{if } i \neq j \end{cases} \tag{3.19}$$

thereby leading to the similarity scores obtained from these probabilities as a *log-odds ratio*:

$$\sigma(i,j) = C \log\left(\frac{q_{ij}}{e_{ij}}\right) \tag{3.20}$$

with C originally set to 2.

To reduce multiple contributions to amino acid pair frequencies from sequences that are too closely related, segments with a sequence identity above a certain threshold are clustered and each cluster is accounted for as a single sequence. Several substitution matrices for different identity thresholds were computed, with the convention that a matrix denoted BLOSUMn corresponds to a n% identity threshold for clustering.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

3.3.2 DNA substitution scores

Although less complex than amino acid similarities and replacement patterns, nucleotide substitution scores can also be calibrated to comply with a certain evolutionary distance, in a PAM-like approach [35].

In practice, one of the following two assumptions are generally made: either all nucleotide mutations occur at equal rates (a scenario adopted in general when dealing with non-coding DNA), or *transitions* are more likely to occur and be accepted than *transversions* due to their silent effect in some cases in coding DNA. Assuming that all 4 nucleotides occur with quasi-equal probabilities, and denoting by κ (usually 3) the transition/transversion ratio, and by α the percentage of mutated bases (which is set to 1%, i.e. 1 PAM), a simple DNA mutation probability matrix can be expressed as

$$M = \begin{pmatrix} 1-\alpha & \frac{1}{\kappa+2}\alpha & \frac{\kappa}{\kappa+2}\alpha & \frac{1}{\kappa+2}\alpha \\ \frac{1}{\kappa+2}\alpha & 1-\alpha & \frac{1}{\kappa+2}\alpha & \frac{\kappa}{\kappa+2}\alpha \\ \frac{\kappa}{\kappa+2}\alpha & \frac{1}{\kappa+2}\alpha & 1-\alpha & \frac{1}{\kappa+2}\alpha \\ \frac{1}{\kappa+2}\alpha & \frac{\kappa}{\kappa+2}\alpha & \frac{1}{\kappa+2}\alpha & 1-\alpha \end{pmatrix} \tag{3.21}$$

where A, C, G, T label the first, second, third and fourth row/column respectively. The matrix of substitution probabilities after n PAMs is obtained by n successive multiplications of M.

Finally, the PAMn nucleotide substitution score results from the *log-odds ratio*

$$\sigma(a,b) = \log\left(\frac{p_a M_{ab}}{p_a p_b}\right) = log(4M_{ab}) \tag{3.22}$$

(since p_a is considered $\frac{1}{4}$ for any nucleotide a). In this manner, one can obtain scores best fit for the conservation rate of the compared sequences. Interestingly, at sufficiently large evolutionary distances (more than 87 PAMs and less than 50% sequence conservation), transition mutations reach a positive score, thereby becoming supporting evidence for sequence homology [35].

Alternatively to this simplified model, nucleotide substitution scores may be derived from real ungapped DNA alignments [38], under a protocol similar to the calculation of the BLOSUM matrices for amino acid substitution. Basically, Chiaromonte et al. [38] extract the data from ungapped alignments of homologous sequences from human and mouse. To exclude strongly conserved regions and thus avoid biases, alignments of sequences that exceed 70% identity are discarded from this analysis. Each aligned pair of nucleotides (a, b) encountered in the data contributes to four counts: the occurrences of (a, b), (b, a), $(complementary(a), complementary(b))$ and $(complementary(b), complementary(a))$, which makes the frequencies both *species symmetric* and *strand symmetric*. The scores are obtained from the observed frequencies of aligned pairs via the classic *log-odds ratio* given by equation (3.1), scaled and rounded to permit integer arithmetic. Three substitution score matrices are proposed (Figure 3.5), derived from regions with different G+C contents, the second matrix being currently used by BLASTZ [12, 184].

3.3.3 Codon substitution models

Substitution probabilities and scores can equally be defined with respect to codons, with the goal of better describing the dynamics of coding DNA. Two notable approaches of deriving codon substitution probabilities exist: the mechanistic approach, which relies on a Markov model of codon substitutions, and the empiric approach, which derives the substitution probabilities from databases of aligned sequences of coding DNA, as it was previously done for computing amino acid similarity scores.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

	A	C	G	T
A	67	-96	-20	-117
C	-96	100	-79	-20
G	-20	-79	100	-96
T	-117	-20	-96	67

	A	C	G	T
A	91	-114	-31	-123
C	-114	100	-125	-31
G	-31	-125	100	-114
T	-123	-31	-114	91

	A	C	G	T
A	100	-123	-28	-109
C	-123	91	-140	-28
G	-28	-140	91	-123
T	-109	-28	-123	100

Figure 3.5: Three nucleotide substitution score matrices proposed in [38], derived from alignments of sequences with 37.4% G+C, 47.5% G+C and 53.7% G+C respectively.

Mechanistic codon substitution models

In [73, 107], codon substitutions are modeled by a Markov model that specifies the relative instantaneous substitution rate from codon i to codon j as:

$$Q_{ij} = \begin{cases} 0 & \text{if } i \text{ or } j \text{ is a stop codon, or} \\ & \text{if } i \to j \text{ requires more than 1 nucleotide substitution,} \\ \pi_j & \text{if } i \to j \text{ is a synonymous transversion,} \\ \pi_j\kappa & \text{if } i \to j \text{ is a synonymous transition,} \\ \pi_j\omega & \text{if } i \to j \text{ is a non synonymous transversion,} \\ \pi_j\kappa\omega & \text{if } i \to j \text{ is a non synonymous transition.} \end{cases} \tag{3.23}$$

for all $i \neq j$. Here, the parameter ω represents the non synonymous-synonymous rate ratio, κ the transition-transversions rate ratio, and π_j the equilibrium frequency of codon j. As in all Markov models of sequence evolution, absolute rates are found by normalizing the relative rates to a mean rate of 1 at equilibrium, that is, by enforcing $\sum_i \sum_{j \neq i} \pi_i Q_{ij} = 1$ and completing the instantaneous rate matrix Q by defining $Q_{ii} = -\sum_{j \neq i} Q_{ij}$ to give a form in which the transition probability matrix is calculated as $P(\theta) = e^{\theta Q}$ [129]. Evolutionary times θ are measured in expected number of nucleotide substitutions per codon.

Empirical codon substitution model

While the mechanistic codon substitution model simulates substitutions with accurate parameters, it does not take into account sufficiently the selective pressure and the resulting effects on the codon conservation. Basically, the mechanistic model focuses on simple mutational events, such as the "third base mutation", which in most cases does not change the encoded amino acid if it is a transition mutation; this is true in some cases of transversion mutations as well.

Nevertheless, there are several other specific conservation families not properly captured by this simulation. A first example is the *aliphatic* conservation (amino acids L, I, V) where corresponding amino acid codons share T at their second base. The last base is, within this group, almost a free choice, while the first has a large degree of freedom. Consequently, it is expected to observe the second T conserved on such codons when aligned with the *aliphatic* group. A similar phenomenon (however with a weaker frequency) appears for the subset (A, S, T) of the small amino acids, where the codons have in common the second base C. In other chemically related amino acid groups, the succession of nucleotide substitutions at the codon level follows more complex paths, as it is the case for positively charged amino acids (R, K), aromatic amino acids (F, Y, W), etc. Such different and complex conservation patterns are difficult to express and model with simple rules. As most of the matrices built for proteins, an empirical estimation gives a better global approximation.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[182] provides the first empirical codon substitution matrix entirely built from alignments of coding sequences from vertebrate. A set of 17,502 alignments of orthologous sequences from five vertebrate genomes yielded 8.3 million aligned codons. The numbers of codon substitutions encountered in this data were used for computing, in a protocol similar to the computation of the popular amino acid similarity scores presented above, 64×64 substitution probability matrices and similarity score matrices for various evolutionary distances.

3.3.4 Statistical significance measures for alignment scores

The alignment score alone does not provide sufficient information for deciding sequence homology, since it needs an interpretation with respect to the results of other alignments in a similar setup. For local alignments, a far more accurate measure of sequence relatedness is given by statistical evaluations of the scores. More precisely, for a score threshold σ, the $P - value$ gives the probability that a local alignment scoring at least σ is found by chance, and the $E - value$ estimates the number of expected local alignments with a score σ or greater that are found by chance.

According to [104], the number of distinct local alignments between two random sequences of length m and n that have a score at least σ is approximately Poisson distributed, with the mean

$$E(\sigma) \approx Kmne^{-\lambda\sigma} \tag{3.24}$$

where the parameters λ and K can be calculated based on the occurrence probabilities of the symbols forming the sequences' alphabet and their similarity scores, as shown in [104]. Also, the optimal alignment score σ approximately follows an extreme-value distribution [81]:

$$P(\sigma' \geq \sigma) \approx 1 - e^{-Kmne^{-\lambda\sigma}}. \tag{3.25}$$

This methodology was introduced in the BLAST [11] software, for expressing the significance of alignment scores between a query sequence and its potential homologs found in a database.

Although there is no rigorous theory supporting a score distribution approximation for gapped local alignments, it was experimentally shown that equations (3.24) and (3.25) can be applied to this case provided that the parameters λ and K are correctly estimated [10].

The estimation method proposed in [162] and further discussed in [10], called the *island method*, is basically an empirical estimation relying on the analysis of sufficiently many local alignments. These alignments are obtained either from two very large sequences, or from several pairs of smaller ones, using a slightly modified Smith-Waterman [188] algorithm that allows memorizing the starting cell of each local alignment within the dynamic programming matrix. In the context of this method, an *island* is then defined as the set of all local alignments starting at the same cell, and the score σ_i assigned to such an island i is the maximum score among the alignments it represents.

Since equation (3.24) is more accurate for larger score values, only islands scoring above a certain threshold c should be considered in order to ensure a good estimate of the statistical parameters [10]. Assuming that there are R_c such islands forming a set denoted I_c, λ_c is estimated to

$$\lambda_c = \ln\left(1 + \frac{1}{\bar{\sigma_c}}\right) + \alpha\left(\frac{1}{m} + \frac{1}{n}\right) \tag{3.26}$$

where m, n are the sizes of the compared sequences, $\bar{\sigma_c}$ is the mean island score in excess of c:

$$\bar{\sigma_c} = \frac{1}{R_c}\sum_{i \in I_c}(\sigma_i - c) \tag{3.27}$$

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

and the term $\alpha\left(\frac{1}{m}+\frac{1}{n}\right)$ is an "edge effect" correction, aiming at reducing biases introduced by considering alignments of limited length, with α chosen as explained in [10]. Depending on λ_c, the estimate for K_c is

$$K_c = \frac{R_c e^{\lambda_c c}}{Nnm} \tag{3.28}$$

with N the number of sequence pairs participating to the comparison and m, n the respective sizes within each pair.

As can be seen from their respective estimation formulas, λ and K are scoring scheme specific, and, introduced in equations (3.24) and (3.25), they give meaning to the alignment scores obtained under that scoring scheme. The obtained values are expected to be very low for related sequences. For instance, a low P-value for an alignment score σ is interpreted as a weak possibility that the compared sequences could be aligned with a score at least σ by chance, enabling one to reject this hypothesis and consider the alignment of score σ as valid homology evidence.

3.4 Seeding techniques for sequence alignment

Section 3.2.3 presented algorithms for finding the optimal alignment between two sequences in quadratic time $\Theta(m, n)$, where m and n are the lengths of the aligned sequences. For values of m and n that have the order of millions, the time necessary to perform the complete calculations cross the acceptable limit, and the optimal answer might be obtained too late to be of any use (for instance, with the current computer power, the exact alignment of two genomic sequences of 100,000,000 characters each is expected to be computed in over 100 years). To overcome this complexity issue, one must relax the requirements and sacrifice optimality in favor of speed.

Most heuristic approaches to finding the alignment of two sequences are based on the assumption that sufficiently related sequences must share at least a well conserved region, called *seed*. The approach basically consists of identifying that region if it exists, and then extending it progressively to the right and to the left as long as the neighboring regions remain sufficiently similar. The optimal alignment of the subsequences identified in this manner can then be constructed with a classic alignment algorithm.

3.4.1 Look-up tables

A key step that contributes to the efficiency of heuristic approaches is the construction of look-up tables that allow instant identification of conserved fragments. The tables are usually based on hashing words of fixed size in conjunction with the use of a linked list connecting all occurrences of the same word in a sequence database [119]. The hashing function generally results from the pattern of matches defined by the seed. For instance, when looking for conserved regions consisting of k consecutive matching symbols in sequences over an alphabet Σ, the hashing function computes for each word w of length k an integer-valued *key* as

$$key(w) = \sum_{i=0}^{k-1} code(w[i]) \cdot |\Sigma|^i \tag{3.29}$$

where $code(\cdot)$ associates an unique value between 0 and $|\Sigma| - 1$ to each symbol in Σ. The formula can be adapted to more complex similarity patterns (such as those described forward in Section 3.4.3 and Section 3.4.5) by ignoring some positions within the word or associating the same code to several elements of Σ.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

3.4.2 First implementations of heuristic approaches: FASTA and BLAST

FASTA/FASTP [130,164] were among the first homology search tools to implement the heuristic approach. They use look-up tables to locate identities or groups of identities between two sequences, typically searching for several groups consisting of k *consecutive matches* ($k = 4$ to 6 in DNA comparison), that are submitted to several subsequent processing steps and may be joined and extended to an alignment depending on their relative locations.

Later on, to respond to the need of aligning very large sequences, BLAST [11] used a more restrictive procedure for identifying conserved regions, consisting in searching a single subsequence of length k identical in both of the compared sequences (k-word), with k significantly larger than the one used in FASTA (typically 11-13 for DNA sequence alignment). Such long identities are expected to be found less often by chance, and are considered a more trustworthy evidence of sequence homology.

For protein alignment, BLASTP uses an approach better suited for the complex similarity relations between amino acids. First, the length k of the k-word considered as a possible core of a similarity region is much smaller in this case (typically 3). Second, instead k-words that are identical in both sequences, BLASTP searches, in one sequence, for a k-word that produces a score higher than a threshold T when aligned with a k-word from the other sequence.

A refinement of BLAST, called Gapped BLAST [13], introduced the notion of *multiple hit* as a compromise between the original FASTA and BLAST approaches. Based on the observation that interesting sequence similarities span much more than the length of a single k-word, the existence of two identity stretches of slightly smaller sizes, on the same diagonal at a relatively short distance, is used instead for locating similarity regions. With the slight disadvantage of space consumption for storing the positions of encountered hits, the approach was shown to improve both the sensitivity and the speed of the search [13].

3.4.3 Spaced seeds

Although the expressiveness of non-contiguous matching patterns in heuristic similarity search had been explored in previous works [29,31,33], *spaced seeds* were first proposed as a concept in the context of DNA sequence alignment by the PatternHunter algorithm [133], and the design of such spaced patterns was discussed in [133] and [31].

Using a spaced seed instead of a contiguous stretch of identical nucleotides to select a potential similarity region has been shown to boost the number of significant alignments that are detected without increasing the rate of random hits [133]. Furthermore, using a seed family, i.e. several seeds simultaneously instead of a single seed, can bring further improvements in this direction [125, 197].

A crucial feature of spaced seeds is their capacity to be adapted to different search situations. Spaced seeds can be *designed* to capture statistical properties of sequences to be searched. An illustrative example is provided by [26, 216] which report on designing spaced seeds adapted to the search of coding regions.

In this section we focus on designing spaced seeds that are appropriate for a class of target alignments with specific statistical properties. We first introduce some general notions that help define the fitness of a seed, and then proceed to discuss models and methods for spaced seed design.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Alignments in the context of heuristic similarity search

Let A be an ungapped alignment of two sequences $S, T \in \Sigma^+$. In the context of heuristic similarity search via spaced seeds, the essential information concerning sequence alignments relies in their succession of matches and mismatches. An alignment A is thereby represented as a sequence over an alphabet $\mathcal{A} = \{0, 1\}$:

$$A = a_0...a_{n-1} \in \{0, 1\}^* \tag{3.30}$$

where 0 marks symbol mismatches and 1 stands for identities [125, 133].

Spaced seed: definition and representation

Definition 2. *A spaced seed π denotes a pattern of required matches and acceptable mismatches in a word of length k, and is generally defined as a subset I_π of positions from $\{0, ..., k-1\}$ which are the required matching positions.*

A spaced seed can be represented more explicitly in form of a string over an alphabet $\mathcal{B} = \{1, *\}$, starting and ending with 1, where the symbol 1 stands for one required match, and $*$ (or "don't care") accepts one match or one mismatch. For a seed π, the following concepts are defined:

Definition 3. *The* **span** *of a seed π, denoted $span(\pi)$, is the length of π's representation as a string over the alphabet $\mathcal{B}$.*

Definition 4. *The* **weight** *of a seed π, denoted $weight(\pi)$, is the the number of 1's appearing in π's string representation, i.e. the number of matching positions required by π.*

Definition 5. *A seed π is said to* **hit** *an alignment $A = a_0...a_{n-1}$ at position $p \in \{0, ..., n - span(\pi)\}$ if and only if*

$$\forall i \in \{0, ..., span(\pi) - 1\}, \pi[i] = 1 \Rightarrow A[p + i] = 1. \tag{3.31}$$

Selectivity and specificity

The efficiency of a seed π is generally expressed by its *sensitivity* and *specificity*.

The *sensitivity* refers to the true positive rate, i.e. the chance that an alignment of interest is hit by the seed at least at one position. More formally,

$$sensitivity(\pi) = \frac{\text{number of alignments of interest that are hit by } \pi}{\text{total number of alignments of interest}}. \tag{3.32}$$

Complementary to the sensitivity, the *specificity* basically denotes the seed's capacity of avoiding random alignments, and is calculated as 1 minus the false positive rate, the latter expressing the chance that an alignment which is not biologically meaningful is hit by the seed:

$$specificity(\pi) = 1 - \frac{\text{number of alignments without biological meaning hit by } \pi}{\text{total number of alignments without biological meaning}}. \tag{3.33}$$

Seeds should be chosen to have an advantageous sensitivity/specificity trade-off. Generally, a less constraining seed, with fewer compulsory match positions, is likely to hit more alignments and achieve a better sensitivity that a seed with more matching positions. However, among the hit alignments, many are likely to be false positives, which makes the seed less selective than a more constraining one. More false positives means more hits that are submitted to a time-consuming similarity evaluation procedure (typically an ungapped or gapped alignment algorithm) that will not produce produce useful results. A good sensitivity/specificity balance is therefore crucial in seed filtration [36], but is generally difficult to achieve.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Lossy seeds vs. lossless seeds

One has to distinguish between the *lossy* and *lossless* cases of seed-based search. These concepts are related to the seed's *sensitivity*.

Lossy seeds are allowed to miss a fraction of target alignments. When designing spaced seeds, the usual goal is to maximize sensitivity over a class of seeds verifying a certain selectivity level.

On the other hand, *lossless* seeds [31] must detect all alignments verifying a given dissimilarity threshold (expressed in terms of a number of errors or a minimal score), and the goal of seed design is to compute a minimal set of seeds with the best selectivity that still ensures the lossless search.

3.4.4 Designing spaced seeds

To address the complex issue of optimal spaced seed design [125, 134, 151], several approaches have been proposed, based on dynamic programming [26,30,105,110], integer programming [210], automata theory [26, 30, 110], or heuristic methods [99, 100].

Generally, the goal of spaced seed design is to obtain spaced seeds or seed families with an advantageous selectivity/specificity trade-off, ensuring effective detection of all or most of the alignments of interest, while minimizing the number of spurious hits. The latter is rather difficult to quantify in practice, but can generally be correlated with the seed's weight: the larger its weight, the less likely is the seed to hit random alignments. On the other hand, the former can be determined with respect to a theoretical model of the target alignments.

In this section, several dynamic programming methods [26, 30, 105, 110] for computing a seed's sensitivity are first discussed. These are generally used in conjunction with sampling, enumerating or otherwise generating spaced seed patterns, and choosing the solution with the highest sensitivity.

Finally, the alternative heuristic approach of [99, 100] is briefly presented, where seed sensitivity is estimated from the overlap complexity of the patterns.

Hit probability computed via dynamic programming on a Bernoulli model

The Keich algorithm [105] was the first dynamic programming approach proposed for computing the probability that a spaced seed π hits a random alignment A of length n generated by a Bernoulli model over the alphabet $\mathcal{A} = \{0, 1\}$ with the probability p of generating the symbol 1 and $1 - p$ of generating 0.

This hit probability is the probability of the union of events $\bigcup_{j=0}^{n-span(\pi)} H_j$ where H_j is the event "π hits A at position j". These events are not independent, since any non-empty subsequence of A influences the presence of hits at several different positions. The approach consists of progressively computing hit probabilities $f(i, b)$ on all prefixes of A with lengths i ranging from $span(\pi)$ to n at all positions j from 0 to $n - span(\pi)$, conditioned by all possible suffixes b of length at most $span(\pi)$: $f(i, b) = P(\bigcup_{j=0}^{i-span(\pi)} H_j : a_{i-|b|}...a_{i-1} = b)$

Informally, the order of calculations of $f(i, b)$ is such that the length i is progressively increased and for each such length the suffix conditioning is progressively relaxed (longer b words are imposed first), *until the value of interest, given by $f(n, \epsilon)$, is obtained*, where ϵ is the empty string.

Each $f(i, b)$ depends on previously computed values of hit probability on a prefix of the word, and on the hit possibilities added by extending that prefix with b. Words b that are not compatible with π are discarded from the calculation, since they do not bring any information:

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

the hit probability is 0 at positions where the presence of 1's does not correspond to the pattern of the seed. Basically, $f(i, b) = 1$ for all compatible strings of length $|b| = span(\pi)$, since the hit is ensured by b regardless of the rest of the word. For all compatible words with $|b| < span(\pi)$, the recursion takes into account the already computed values corresponding to the same alignment length i and conditioned by a longer suffix. Longer suffixes are obtained by prefixing b with either 0 or 1, each of these having a probability dictated by the underlying Bernoulli model (i.e. $1 - p$ and p respectively). Hence, $f(i, b) = (1 - p)f(i, 0b) + pf(i, 1b)$. Note that, while $1b$ is always compatible with π if b is compatible, meaning that $f(i, 1b)$ can be found in the dynamic programming table, it is not always the case for $0b$. If $b' = 0b$ is not compatible, then it prevents any hits from occurring at position $i - span(\pi)$. In this case the hit probability $f(i, b')$ is no different from $f(i - 1, b' \gg 1)$, where $b \gg j$ is obtained from b by removing the last j symbols. $f(i - 1, b' \gg 1)$ can either be found in the table if $b' \gg 1$ is compatible with π, or can be computed recursively by the same reasoning otherwise.

This approach is adapted in [26] to alignments generated by a more complex model, in order to obtain a seed's sensitivity with respect to alignments of coding regions modeled by HMMs.

Approaches based on automata theory

Seed automaton Considering the formalization of the concept of *hit* given by definition 5, it is useful to think of the symbols in the alphabet $\mathcal{B}$ of the seed pattern as representatives of a certain subset of $\mathcal{A}$ (the alphabet of ungapped alignments), which they accept. This association is the basis of an extension of spaced seeds, called *subset seeds* [110], which we briefly discuss in Section 3.4.5. In the case of spaced seeds, the symbol $1 \in \mathcal{B}$ represents (and thereby accepts) the subset $\{1\}$ of $\mathcal{A}$, and the symbol $* \in \mathcal{B}$ represents and accepts the subset $\{0, 1\}$ of $\mathcal{A}$. From this perspective, it is easy to perceive a seed π as a regular expression defining the acceptable matching patterns. For example, a seed $\pi = 11*1$ describes the words of length 4 with 1's at their first, second and last positions ($I_\pi = \{0, 1, 3\}$), which are captured by the regular expression $11(0|1)1$. Moreover, according to equation (3.31), all alignments containing one of this class of words are hit by the seed π. All alignments hit by π are therefore captured by the regular expression $(0|1)^*11(0|1)1(0|1)^*$, which basically describes the language of words over the alphabet $\mathcal{A}$ containing the pattern of the seed π at some arbitrary position.

Since, for any regular expression, there is a finite state automaton that defines the same language [95], the complete set of alignments recognized by a seed can be represented as an automaton. In the following, we give some formal definitions for several automata-related concepts.

Definition 6. *A deterministic finite automaton (DFA) D is defined by a 5-uple $(Q, \Sigma, \delta, q_0, F)$ where:*

- *Q is a finite set of states;*
- *Σ is a finite state of input symbols;*
- *$\delta : Q \times \Sigma \to Q$ is a transition function which switches to a new state in Q given a current state and an input symbol;*
- *$q_0 \in Q$ is the initial state;*
- *$F \subseteq Q$ is a set of* final *or* accepting *states.*

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

The function δ can be extended to words in the form of $\hat{\delta} : Q \times \Sigma^* \to Q$ whose definition is based on successive applications of δ:

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q \\ \hat{\delta}(q, wa) &= \delta(\hat{\delta}(q, w), a) \end{aligned} \tag{3.34}$$

where ϵ is the empty string, $a \in \Sigma$ is an input symbol and $w \in \Sigma^*$ is a word on the alphabet of input symbols.

Definition 7. *The language accepted by a DFA $D = (Q, \Sigma, \delta, q_0, F)$ is defined by*

$$L(D) = \{w | \hat{\delta}(q_0, w) \in F\}. \tag{3.35}$$

More general than deterministic automata, non-deterministic automata can have several current states simultaneously. The main difference with respect to the deterministic case is the transition function δ which, given a current state and an input symbol, returns a subset of Q that can contain more than one state.

Definition 8. *A non-deterministic finite automaton (NFA) N is defined by a 5-uple $(Q, \Sigma, \delta, q_0, F)$ where:*

- *Q is a finite set of states;*
- *Σ is a finite state of input symbols;*
- *$\delta : Q \times \Sigma \to 2^Q$ is a transition function which associates to a current state and an input symbol a possibly empty subset of Q;*
- *$q_0 \in Q$ is the initial state;*
- *$F \subseteq Q$ is a set of* final *or* accepting *states.*

Definition 9. *The language accepted by a NFA $N = (Q, \Sigma, \delta, q_0, F)$ is defined by*

$$L(N) = \{w | \hat{\delta}(q_0, w) \cap F \neq \emptyset\}. \tag{3.36}$$

For some languages (including the language of alignments accepted by a spaced seed) a NFA is easier to construct than a DFA. Nevertheless, every NFA has an equivalent DFA that accepts the same language, and every DFA has an equivalent NFA [96]. An NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ can be transformed in its equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ with the same input alphabet Σ by *subset construction* [96]:

$$\begin{aligned} Q_D &= 2^{Q_N} \\ q_{0D} &= \{q_0\} \\ F_D &= \{S \in Q_D | S \cap F_N \neq \emptyset\} \\ \delta_D(S, a) &= \bigcup_{q \in S} \delta_N(q, a), \text{ with } \delta_D : Q_D \times \Sigma \to Q_D. \end{aligned}$$

Since Q_D is the power set of Q_N, the number of states of a DFA can be exponential relatively to the number of states of its NFA equivalent. However, in practice, some states in this power set are not accessible from the initial state and can therefore be ignored. Moreover, this construction

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

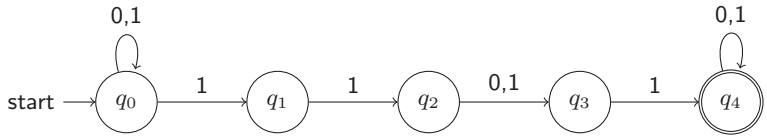

Figure 3.6: Non-deterministic finite-state automaton describing the language of alignments accepted by the seed 11*1.

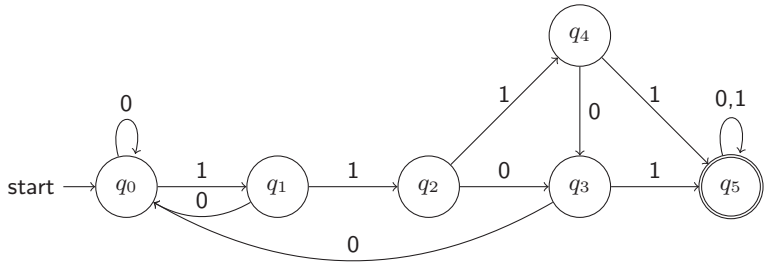

Figure 3.7: Deterministic finite-state automaton describing the language of alignments accepted by the seed 11*1, equivalent to the non-deterministic finite-state automaton of Figure 3.6.

does not guarantee the resulting DFA is defined without redundancies, and some states may be *equivalent*. The equivalence relation of two states q_i and q_j is defined formally as

$$\forall w \in \Sigma^*, \hat{\delta}(q_i, w) \in F \Leftrightarrow \hat{\delta}(q_j, w) \in F. \tag{3.37}$$

Being an equivalence, this relation is transitive [97], which allows sets of equivalent states to be replaced by a single state without any effects on the language defined by the automaton. Reducing the number of states of an automaton is called *minimization*, and several methods exist for performing this transformation [94, 146].

The NFA of the seed 11*1 is depicted in Figure 3.6, which makes use of the classic directed graph representation of automata, where states are nodes and transitions are arcs labeled with the respective input symbols. Its construction follows easily from the aforementioned regular expression describing the alignments recognized by the seed: $(0|1)^*11(0|1)1(0|1)^*$. Generally, any NFA $N_\pi = (Q, \mathcal{A}, \delta, q_0, \{q_{n-1}\})$ corresponding to a seed π will have a linear shape, with a number of states $n = 1 + span(\pi)$, with looping transitions in its initial state q_0 and its only final state q_{n-1} for all input symbols in $\mathcal{A}$ (the alignment alphabet), and with transitions between states q_i, q_{i+1} labeled by the subset of $\mathcal{A}$ accepted by the symbol at position i in the seed pattern. The equivalent DFA is depicted in Figure 3.7.

The concept can be easily extended for multiple spaced seeds (spaced seed families) in order to express the complete set of alignments recognized by at least one member of the family, as depicted in Figure 3.8.

We have chosen to represent seeds by NFA automata because obtaining the structure of the NFA from the seed pattern is straightforward. However, in practice, an equivalent DFA is used instead due to its deterministic functioning. Moreover, the subset construction of the seed DFA from the corresponding NFA is not used in the seed design approaches we discuss below [30, 108]. Instead, the DFA is obtained directly using specific methods.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

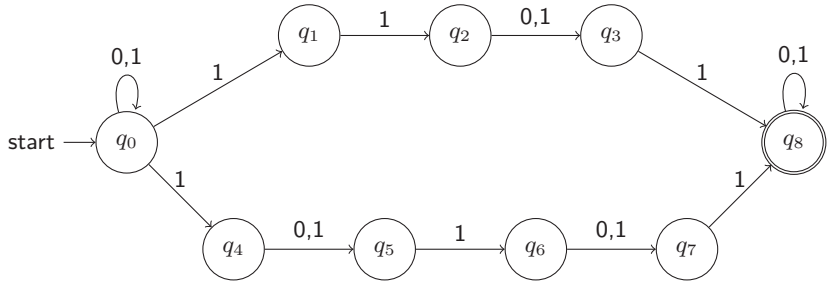

Figure 3.8: Non-deterministic finite-state automaton describing the language of alignments accepted by the seed family $\{11^*1, 1^*1^*1\}$.

Construction of the spaced seed DFA using the Aho-Corasick algorithm In [30], Buhler et. al. propose a method based on the Aho-Corasick algorithm [3] for constructing the DFA that accepts all the alignments recognized by a spaced seed.

Given a spaced seed π of weight w and span s, and W_π the set of all the 2^{s-w} strings accepted by the seed π, the algorithm consists of transforming the trie constructed with W_π into a DFA $D_\pi = (Q, \{0,1\}, \delta, q_0, F)$ that accepts all the strings of arbitrary length larger or equal to s, containing a member of W_π. First, every node in the trie becomes a state in D_π, the root of the trie corresponding to the initial state, and its leaves to final states. The arcs in the trie become transitions in D_π. Considering that the nodes (and, in consequence, the states) are labeled with strings corresponding to the path followed in the trie from the root to reach them, transitions corresponding to *failure links* are added, from any state labeled with a string S to the state labeled with its longest proper suffix (which can be empty). Hence, whenever a 0 symbol in an input string prevents from following the trie, a 0 labeled transition allows the search to continue with its largest compatible suffix.

At this point, the Aho-Corasick algorithm constructed a DFA that accepts all the words of arbitrary length over the alphabet $\{0,1\}$ which end with a word from W_π. To ensure that all words having a substring from W_π at any position are accepted, all final states q_f are transformed into absorbing states, i.e. transitions to themselves are added for all input symbols. Finally, since all final states are obviously equivalent, they can be compacted into a single one.

The automaton obtained with the above process for the seed 1*11 is illustrated in Figure 3.9. The algorithm does not guarantee to obtain the minimal DFA.

A more compact spaced seed DFA construction In [111], a method for constructing the DFA for *subset seeds* (briefly presented later in Section 3.4.5) is proposed. Since subset seeds are an extension of spaced seeds, this construction directly applies to spaced seeds.

Let $\pi \in \{1,*\}^*$ be a spaced seed of span s and weight w, R_π the set containing the positions of *'s in the seed pattern, with $|R_\pi| = s - w = r$, and $D_\pi = (Q, \{0,1\}, \delta, q_0, \{q_f\})$ the DFA defining the language of alignments recognized by π. The states of D_π are labeled with pairs $\langle X, t\rangle$, $X \subseteq R_\pi$, $t \in \{0, ..., s\}$, such that, for any input $a_1...a_p$ reaching the state $\langle X, t\rangle$, t is the largest value $\leq s$ to verify $a_{p-t+1}...a_p = 1^t$, and X contains all $x_i \in R_\pi$ such that a $\pi_1...\pi_{x_i}$ matches a suffix of $a_1...a_{p-t}$. This labeling helps to determine, at the DFA construction step, transitions similar to *failure links*, i.e. the largest prefix of the seed that matches a suffix of the input.

The initial state of this automaton is labeled $\langle \emptyset, 0\rangle$. The transition function $\delta : Q \times \{0,1\} \to Q$ is defined by:

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

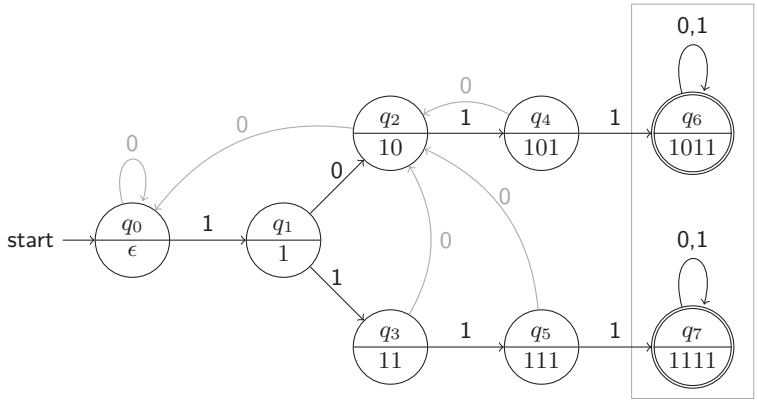

Figure 3.9: DFA constructed using the Aho-Corasick algorithm for the spaced seed 1*11, accepting the alignments $(0|1)^*1(0|1)11(0|1)^*$. Each state is labeled with the string corresponding to the paths in the trie that led to it from the root. The black transitions correspond to paths in the trie of the set $\{1011, 1111\}$. The transitions in grey correspond to *failure links*. The accepting states are equivalent and can be compacted into a single state.

- $\delta(\langle X, t\rangle, 1) = \langle X, t+1\rangle$;
- $\delta(\langle X, t\rangle, 0) = \langle \{x \in R_\pi | x \leq t+1\} \cup \{(x+t+1) \in R_\pi | x \in X\}, 0\rangle$.

All states $\langle X, t\rangle$ with $max(X)+t = s$ are final and equivalent, and can be compacted in a single state q_f.

This method was shown to construct automata that have a number of states smaller than or equal to the Aho-Corasick automaton described above.

Seed sensitivity in a Markov model of ungapped alignments A dynamic programming algorithm for computing the probability that an automaton $D_\pi = (Q, \{0,1\}, \delta, q_0, \{q_f\})$ accepts a random alignment of length n generated by a kth-order Markov model is proposed in [30]. A kth-order Markov model M describes a discrete stochastic process with a finite set of states Q_M, each state having a certain probability to be the initial state, where the transitions to a state $s \in Q_M$ at any given step t is conditioned by the previous k steps: $P_M(s_t = s | s_{t-1} = s_{i_1}, ..., s_{t-k} = s_{i_k})$. For this particular case, Q_M coincides with $\{0,1\}$, and the succession of states at n consecutive steps describes a string A of length n over the alphabet $\{0,1\}$ of ungapped alignments.

Let $\alpha \in \{0,1\}^k$ be a string of length k. For any state $q \in Q$ of the automaton D_π and any $b \in \{0,1\}$, let $\delta^{-1}(q,b) = \{q' \in Q : \delta(q', b) = q\}$, be the set of states that transition to q on the input b. The probability that the automaton D_π reaches state q after reading an input $A \in \{0,1\}^t, A = a_1...a_t$, with the last k steps of A corresponding to the history αb, is recursively defined as

$$P(q, t, \alpha b) = P_M(a_t = b | a_{t-1}...a_{t-k} = \alpha) \cdot \sum_{q' \in \delta^{-1}(q,b)} \sum_{b' \in \{0,1\}} P(q', t-1, b'\alpha) \tag{3.38}$$

where $P_M(a_t = b | a_{t-1}...a_{t-k} = \alpha)$ is the transition probability given by M. For inputs shorter than k, the history is truncated to the last t steps. The initialization sets $P(q_0, 0, \epsilon) = 1$ for the initial state, and $P(q, 0, \epsilon) = 0, \forall q \in Q \backslash \{q_0\}$. The calculation stops at step $t = n$, and the seed

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

sensitivity is the probability that D_π has reached its final state q_f:

$$sensitivity(\pi) = \sum_{\beta \in \{0,1\}^n} P(q_f, n, \beta). \tag{3.39}$$

Seed sensitivity in any probabilistic model In [110], a more general methodology based on the finite automata theory is proposed for designing spaced seeds. The central idea is to model the set of target alignments by a *finite-state probability transducer*, i.e. finite automata without final states whose transitions output probabilities, which can basically express any of the aforementioned probabilistic models for biological sequences.

Definition 10. *A probability transducer G over an alphabet $\mathcal{A}$ is a 4-uple $(Q_G, \mathcal{A}, \rho_G, q_0)$, where:*

- *Q_G is a finite set of states;*
- *$\rho_G : Q_G \times \mathcal{A} \times Q_G \rightarrow [0,1]$ is a probability function such that $\forall q \in Q_G, \sum_{q' \in Q_G, a \in \mathcal{A}} \rho_G(q, a, q') = 1$;*
- *$q_0 \in Q_G$ is an initial state.*

The triplets $e =< q, a, q' >$ with non-zero probabilities define transitions in the transducer, and each such transition is labeled by its symbol $a \in A$. In deterministic transducers, there is at most one transition $< q, a, q' >$ for any $a \in A$ and any $q \in Q$.

A *path* $P = (e_1, ..., e_n)$ in G, labeled by the concatenation of the symbols a_i corresponding to each transition, is called *initial* if it starts in the state q_0, and has the associated probability

$$\rho(P) = \prod_{e_i \in P} \rho(e_i). \tag{3.40}$$

More generally, the probability of a word w, $\mathcal{P}(w)$ is the sum of all probabilities $\rho(P)$ of initial paths labeled with w. Finally, the probability of a language L is defined as

$$\mathcal{P}(L) = \sum_{w \in L} \mathcal{P}(w). \tag{3.41}$$

The sensitivity of a seed π accepting a set (language) of alignments L_π defined by the automaton D_π, with respect to a set (language) of target alignments L_T defined by the automaton D_T, and a probability transducer G is given by the conditional probability

$$\frac{\mathcal{P}_G(L_\pi \cap L_T)}{\mathcal{P}_G(L_T)}. \tag{3.42}$$

The language intersection $L_\pi \cap L_T$ is the language defined by the product of D_π and D_T. In order to compute the probabilities in (3.42) as shown by (3.41), a *probability-weighted automaton* is computed as the product between the automaton defining the language of interest and the transducer. The result of this product can be seen as a non-deterministic probability transducer with final states. More formally, the product between an automaton $K = (Q_K, \{0,1\}, \delta_K, q_K^0, F_K)$ defining the language L_K and a probability transducer $G = (Q_G, \{0,1\}, \rho_G, q_G^0)$ is $W = (Q_W, \{0,1\}, \delta_W, q_W^0, F_W)$ with:

- $Q_W = Q_K \times Q_G$;

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

- $q_W^0 = (q_K^0, q_G^0)$;
- $F_W = \{(q_K, q_G) | q_K \in F_K\}$;
- $\delta_W((q_K, q_G), a, (q'_K, q'_G)) = \begin{cases} \rho_G(q_G, a, q'_G), & \text{if } \delta_K(q_K, a) = q'_K \\ 0, & \text{otherwise.} \end{cases}$

Final states in this probability-weighted automaton correspond to accepting states in the automaton K, meaning that any input reaching such a final state belongs to the language accepted by K. In consequence, the probability of the language L_K with respect to the transducer G can be computed as

$$\mathcal{P}_G(L_K) = \sum_{P:P \text{ is a full path in } W} \rho(P) \tag{3.43}$$

where a full path is an initial path ending in a final state.

The method is implemented in the IEDERA software [110–112] that uses the above algorithm to explore the space of possible seeds and select most sensitive seeds using a sampling procedure for seeds and respective hit positions and by performing a local optimization on the best candidates.

Heuristic optimization of multiple spaced seeds

In [100], a heuristic approach for optimizing multiple spaced seeds is proposed. The method relies on the overlap complexity, which is empirically shown to be well correlated with sensitivity, but much easier to compute. Intuitively, good seeds should have a low number of overlapping hits, because overlapping hits generally detect the same similarity regions.

Given two spaced seeds π_1 and π_2, the number of non-overlapping hits is estimated by first computing the values $\sigma(i)$, the number of 1's aligned together when a copy of π_2 shifted by i positions is aligned against π_1, for all (possibly negative) i ranging from $1 - |\pi_2|$ to $|\pi_1| - 1$. Then, the overlap complexity for two seeds is defined as

$$OC(\pi_1, \pi_2) = \sum_i 2^{\sigma(i)}. \tag{3.44}$$

The overlap complexity of a seed family $\{\pi_1, ..., \pi_k\}$ is obtained as the sum of overlap complexities for all possible pairs of seeds $OC(\pi_i, \pi_j), 0 \leq i \leq j < k$.

With this measure for approximating the sensitivity, a polynomial heuristic algorithm for designing multiple spaced seeds is then proposed. Basically, the method consists of starting with fixed seeds and repeatedly modifying them (changing 1's into *'s) as long as the overlap complexity improves. This approach has the advantage of being very fast while also giving good results in general. Although no explicit mechanism is proposed for correlating the seed patterns with a specific model (mismatch distribution) in the target alignments, the method could be refined in this direction by introducing in (3.44) different weights for the overlapping seed positions.

3.4.5 Other types of seeds

The methods for detecting the well conserved regions of two sequences have evolved beyond contiguous and spaced matching patterns, leading to the emergence of more complex models [28].

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Vector seeds [27] were proposed as a generalization of spaced seeds. Such a seed π consists in a vector v of l integers and a score T, with the meaning that π hits an alignment $A = a_0...a_n$ if there is a position i in this alignment such that $v \cdot (a_i, ..., a_{i+l-1}) \geq T$. Both contiguous and spaced seeds can easily be defined as vector seeds (v, T) by making v a vector whose length equals the seed's span and has a 1 for every position corresponding to a match and a 0 for every position corresponding to a mismatch in the seed pattern, and finally setting the threshold T to be the weight of the seed. Although very powerful in theory for expressing local similarities, vector seeds are not always compatible with direct hashing methods and require rather complex and time consuming approaches for seed detection.

Neighbor seeds [48], initially named *daughter seeds* [47], were proposed as an approximation of vector seeds. In conjunction with a *two step extension* approach which consists of extending a hit only if other matching positions are found besides the ones defined by the seed pattern, daughter seeds are essentially multi-pass filters based on spaced seed patterns which allow the expressiveness of vector seeds to be achieved while using an efficient hash-based index implementation, and giving the possibility to maximize independence between seeds, which is known to improve the overall sensitivity.

Subset seeds [108], another extension of spaced seeds, combine the expressiveness of an arbitrary alignment alphabet with the advantages of using direct hashing [119] for locating seeds. Basically, the target alignments are words over an alphabet $\mathcal{A}$ containing a symbol 1 which denotes exact matches and other symbols for different mismatch classes. The seed pattern is defined over an alphabet $\mathcal{B}$, where each symbol in $\mathcal{B}$ accepts a set of symbols (types of mismatches) from $\mathcal{A}$, with the additional constraint that every symbol in $\mathcal{B}$ accepts matches (its corresponding subset of $\mathcal{A}$ necessarily contains the symbol 1) and there is a symbol in $\mathcal{B}$ accepting *only* matches.

In practice, the subset seeds for DNA alignment generally distinguish between transitions and transversions mismatches [156, 215], while in proteins [177] the set of amino acids can be partitioned in various similarity groups based on their physical and chemical properties.

Indel seeds [135] are a seed models which explicitly allow indels, and were designed with the primary goal of boosting the sensitivity of heuristic search in non-coding DNA, especially in repeats where insertions and deletions are relatively common. The alignment alphabet $\mathcal{A} = \{0, 1, 2, 3\}$ contains, in addition to the usual symbols 1 for matches and 0 for mismatches, two symbols standing for insertion and deletion respectively. In turn, indel seeds are patterns over the alphabet $\{1, *, \times\}$, where $\times$ accepts all possible mutational events, indels included. Applying an indel seed to a sequence means selecting not just one subset of positions as it is done, for instance, in the case of spaced seeds, but up to 3^k different sets, where k is the number of $\times$ symbols in the seed pattern.

As reported by their respective authors, most theoretical aspects discussed for spaced seeds, i.e. the evaluation of seed sensitivity based on the automata theory, apply to the alignment and seed models presented above, with the additional complexity resulting from a more refined alphabet for both the target alignments and the seed pattern.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Part II

Frameshift mutation discovery via protein back-translation

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 4

Context and motivation

4.1 Problem statement

Gene duplication and divergence provide a critical source of genetic novelty during evolution [4]. It is believed that, after duplication, there is basically no selection pressure for maintaining both copies in an identical functional state, since one copy suffices for accomplishing the function [4]. Hence, one of the copies may undergo loss-of-function mutation that can remain uncorrected without affecting the functioning of the organism. However, it is not unlikely that both copies remain functional, while their sequence and expression patterns diverge and they assume different roles within the organism. This hypothesis may actually explain large families of genes with related functions in complex organisms [4].

Here, we address the problem of finding distant homologies, in particular when *frameshift events* are involved in the divergence. We consider the following "duplication and divergence scenario". A gene encoding a protein is duplicated. One of the copies undergoes a frameshift mutation. Subsequently, both copies may be affected by point mutations, mostly synonymous for one of them in order to preserve the function, but unrestricted for its frameshifted copy.

Tracing back the common origins of the two sequences may become difficult after a long period of evolution. As mentioned earlier in Section 1.4, a single frameshift mutation introduces a drastic change in the translated protein sequence, completely changing the translation after the frameshift (Figure 4.1), while having a small effect on the affected DNA sequence, most of which is left unchanged. However, if additional point mutations are involved in the divergence, the similarity at the DNA level may be reduced beyond recognition. It has been shown [76, 80, 187] that, in coding DNA, there is a base compositional bias among codon positions, that no longer applies after a reading frame change. A frameshifted coding sequence can be affected by base substitutions leading to a composition that complies with this bias. If, in a long evolutionary time, a large number of codons in one or both sequences undergo such changes, they may be altered to such an extent that the common origin becomes difficult to observe by direct DNA comparison. Figure 4.2 illustrates such an extreme case. The sequences from Figure 4.1 were altered by synonymous mutation following the frameshift. Note that the protein sequences remain unchanged when compared to Figure 4.1, but the mutated DNA presents an impressive number of mismatches, making it difficult for a sequence alignment algorithm to detect any similarity.

In conclusion, such divergence scenarios may become difficult to detect both at the DNA and protein sequence level. Here, we discuss the design of a procedure that combines DNA and protein sequence information in order to improve homology detection.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

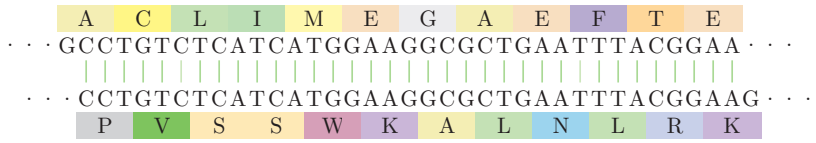

Figure 4.1: Proteins encoded on identical DNA sequences, with a frameshift.

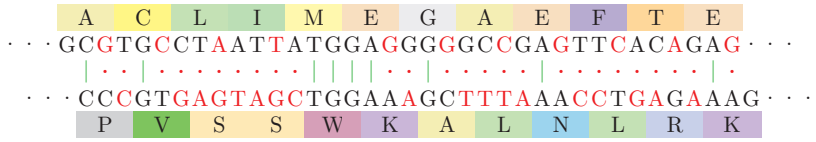

Figure 4.2: The same proteins, encoded on sequences that were identical at one point, but subsequently underwent a series of synonymous mutations.

4.2 Related work

The problem of detecting frameshift mutations that led to functional divergence has so far known several approaches. Several studies [82, 83, 159, 172] assumed sufficient similarity at the DNA level and used the straightforward solution given by BLAST [11, 13] alignment approach: either BLASTN on DNA and mRNA, or TBLASTN on mRNA and proteins. Although capable of insightful results thanks to the six frame translations, TBLASTN addresses the problem of frameshifts only indirectly and in a heuristic manner. TBLASTN uses a greedy algorithm to link nearby distinct alignments and computes an E-value for these linked sets [9]. However, frameshifts are not labeled as such by TBLASTN, and the only evidence that the sets are correctly linked is that they are in a consistent order and yield a lower E-value than each alignment would if taken on its own [67].

Interestingly, years before the publication of these results, various algorithmic methods specific to frameshift discovery had already been proposed.

4.2.1 Amino acid substitution score matrices for frameshift detection

For *handling frameshifts at the protein level*, [40] and [167] propose to use 5 substitution score matrices, each capable of detecting a different frameshift: insertion, deletion, and inversion in 3 reading frames. These scores are obtained as a classic log-odds ratio

$$\sigma(i,j) = \frac{1}{\lambda} \log \left(\frac{q_{ij}}{p_i p_j} \right) \tag{4.1}$$

with q_{ij} the probability that i and j have evolved from a common ancestor and p_i the probability of occurrence of i, and can thereby be integrated directly with regular alignment tools, such as BLASTP. Both approaches are based on evolution scenarios involving gene duplication, followed by a frameshift in one of the copies, and subsequent point mutations and in-frame insertions and deletions in both sequences. An alternative scenario is proposed in [40] for simulating the occurrence of a sequencing error: the frameshift is more recent and affects one of the sequences after both already drifted by point mutations.

Since the hypothetical evolution process has several distinct stages, different evolution models are considered for each stage and combined to obtain the joint probabilities q_{ij}.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

More precisely, [40] proposes to compute the probabilities q_{ij} as

$$q_{ij} = p_i \sum_k M_{ik} \sum_l T_{kl} M_{lj} = p_i (MTM)_{ij} \tag{4.2}$$

where M_{ik} gives the $i \rightarrow k$ substitution frequency according to the point mutation process (in practice, PAM models), and T_{lk} is the probability of l turning into k due to a certain type of frameshift (e.g. one base insertion). The latter depends on the genetic code and the codon frequencies, and takes into consideration codons on different reading frames that have at least two consecutive nucleotides in common. For example, the deletion of a nucleotide can trigger transformations such as $C_1C_2C_3 \rightarrow C_2C_3N$, where $C_1C_2C_3$ is a codon, with $C_i \in \{A, C, G, T\}$, and N denotes any nucleotide. Five theoretical T models (one for each of the five different frameshifts mentioned above) are derived from these codon transformations by shifting, in conjunction with the known codon frequencies. The resulting scoring matrices are asymmetric, due to the asymmetry of T_{lk}.

In the model proposed in [167], point mutation rates are retrieved from the BLOSUM [86] substitution scores, by "reverse-engineering" equation (3.20) (Chapter 3, Section 3.3):

$$P_B(i \rightarrow j) = p_j 2^{BLOSUM62_{ij}/2} \tag{4.3}$$

with p_j chosen so that $\sum_i P_B(i \rightarrow j) = 1$. The effect of the frameshift is captured in a separate matrix P_{fs}, which accounts for the conversion of a particular codon for amino acid i into amino acid j in a different reading frame, based on the same reasoning as above,

$$P_{fs}(i \rightarrow j) = \frac{p_{fs}(i, j)}{p_{fs}(i)} \tag{4.4}$$

where $p_{fs}(i, j)$ is the probability of finding the (i, j) amino acid pair and $p_{fs}(i)$ is the probability of finding the amino acid i. The possible effects of codon bias are not taken into account. Finally, the full frameshift substitution probability matrix must capture the fact that a frameshift is followed in both sequences by point mutation, and is therefore computed as

$$P_{tot}(i \rightarrow j) = P_B^n(i \rightarrow k) P_{fs}(k \rightarrow l) P_B^n(i \rightarrow j) \tag{4.5}$$

where P_B^n is the nth power of P_B^n, allowing one to obtain scores that comply with a certain evolutionary distance. This probability matrix is then converted to log-odds scores according to the classic formula:

$$\sigma(i, j) = log \frac{p_i P_{tot}(i \rightarrow j)}{p_i p_j} = log \frac{P_{tot}(i \rightarrow j)}{p_j}. \tag{4.6}$$

As mentioned above, these scoring matrices have the advantage of being ready to use with classic sequence alignment methods, such as BLAST or any dynamic programming aligner based on the Smith-Waterman algorithm. On the downside, multiple reading frame changes within the same alignment (for example, if two sequences start on the same reading frame and a frameshift occurring in the middle of one of the sequences switches the reading frame until its end) are difficult to manage when using these regular alignment methods.

Another disadvantage, this time of the scores themselves, is the partial loss of information. Frameshifts are considered independently with respect to each amino acid in the sequence, taking into account the largest codon part that was preserved. The scoring scheme does not capture the fact that, after the reading frame change, the codons following the frameshift are

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

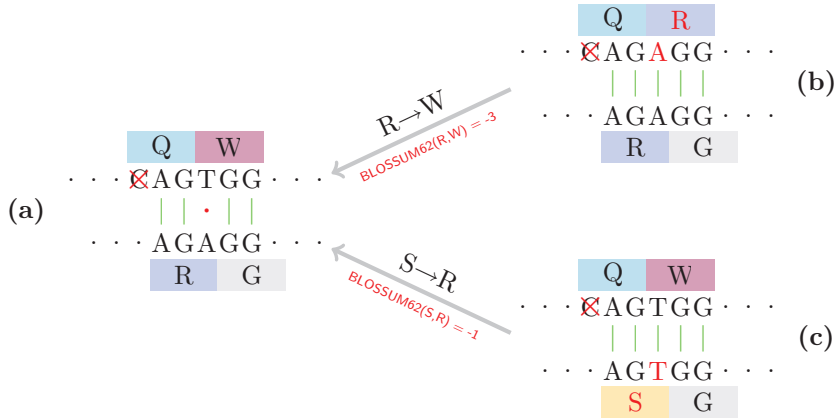

Figure 4.3: Illustration of an alignment (a) with an overestimated score under the scoring schemes of [40] and [167]. Since only the largest codon overlaps (2 positions) are taken into account by the scores, while the smaller overlaps (of one codon position) are ignored, the score of this alignment is likely to be overestimated. The pairs (Q,R) and (W,G) have high substitution scores respectively, because they share two consecutive nucleotides on different reading frames. Nevertheless, the overlap between the last codon position of R (a purine) and the first codon position of W (a pyrimidine) is not taken into account by the scoring system. Assuming evolution of the sequences by point mutations, if these sequences share a common ancestor, then some purine-pyrimidine mutation should have taken place during the evolution, either on the first position of the codon of W (b), or on the last position of the codon of R (c). However, both these mutations are improbable, as suggested by their negative BLOSUM62 scores.

composed of parts from two consecutive codons from the original coding sequence. For instance, a deletion ($C_1C_2C_3 \rightarrow C_2C_3N$) produces codons whose last nucleotide was the first nucleotide of the codons in the original coding sequence, prior to the frameshift. If considered in the context of *two consecutive codons*, the deletion should be expressed as $C_1C_2C_3\ C'_1C'_2C'_3 \rightarrow C_2C_3C'_1\ C'_2C'_3N$, constraining the choice of amino acids that should have a high score in an alignment when paired with the amino acid encoded by $C_1C_2C_3$ to those compatible with C'_1 on the last position of their codons. While this compatibility is not an issue for the amino acids encoded by codons with all possible nucleotides on their third position, as they are not constrained by the following codon $C'_1C'_2C'_3$, ignoring the first position/third position shifting is a rough approximation in general. Take, for example, the amino acid Q, encoded by the codons CAA and CAG. Under the assumption of a deletion frameshift on the first codon position ($C_1C_2C_3 \rightarrow C_2C_3N$), the amino acids whose codons start with AA and AG should be obtained, namely S, R, K and N. To obtain R or K, a codon starting with a purine must follow the one encoding the original Q, and to obtain S or N the next codon must start with a pyrimidine. Even assuming evolution by point mutations, some purine/pyrimidine substitutions on the first or last codon position are unlikely since they trigger substitutions between amino acids with different characteristics. It is the case of amino acids substitutions such as $W \leftrightarrow R$, $W \leftrightarrow G$, $Y \leftrightarrow D$, $P \leftrightarrow T$, etc. Differences between their respective physical and chemical properties have been illustrated earlier in Figure 1.8 of Chapter 1, Section 1.2. Additionally, their low substitution rates are reflected in the negative scores given by classic amino acid substitution matrices such as BLOSUM62. An example of score overestimation issues that arise from ignoring the double codon overlap in the scoring scheme when performing frameshifted alignments directly at the protein level is given in Figure 4.3.

The following chapter will describe a way to reduce the approximation and the information

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

loss resulted here from handling frameshifted amino acids independently, in addition to flexible handling of multiple reading frames within the same alignment.

4.2.2 Dynamic programming alignment algorithms that handle frameshifts

Several dynamic programming approaches have been proposed for handling frameshifts in DNA-protein or DNA-DNA pairwise alignment [16,79,85,106,166,168,214], which are, to some extent, adaptations of the local alignment algorithm [188].

Based on the idea that proteins evolve slower than DNA, these algorithms make use of the translation of the DNA sequence to determine homologies or detect off-frame sequencing errors. In consequence, aligned sequences are generally handled in groups of consecutive symbols denoting codons. These groups are of size 3 if no gaps exist within the codon, but segments of size 1 2, 4 or 5 are considered for more flexible placements of gaps. Scoring the alignment between two such segments, between a segment and an amino acid is based on the amino acids similarity scores provided by popular matrices such as BLOSUM62 in conjunction with the segment's possible translations as an amino acid. When multiple translations are possible, as is the case for segments of sizes different than 3, where symbols need to be added or removed to obtain a standard codon, the translation resulting in the best pairwise score is the translation of choice. In general, different reading frames are handled in different dynamic programming tables, and a frameshift is marked in a partial alignment whenever a switch to a different reading frame yields a better score in a partial alignment.

The formalizations of these methods are rather complex, since they involve segment alignments, translations, and specific gapping rules. Nevertheless, they rigorously model the evolution of coding DNA and provide a way to handle mutations *in their context* rather than uniformly at any point of the coding sequence.

4.2.3 Protein back-translation and alignment of *sequence graphs*

Back-translation (or reverse translation) of a protein usually refers to obtaining one of the DNA sequences that encodes the given protein. Several methods for achieving this exist [74, 193], aiming at finding the DNA sequence that is most likely to encode that protein. Some programs use multiple protein alignments to improve the back-translation [72, 147]. This concept can be considered opposite to the "translation way", where translation is used to improve coding DNA alignments or assess new coding DNA [22, 62, 168, 198, 209].

From a different perspective, instead of being a goal, back-translation can be used as a powerful instrument in the context of protein homology discovery. The back-translation of amino acids in a protein sequence into their respective codons and comparisons at the codon level can provide more flexibility to sequence alignments.

For example, the score matrices mentioned in Section 4.2.1 make use of the amino acids' back-translation in order to infer rates of mutations caused by frameshifts. Marginally related to this idea, yet not designed for dealing with frameshifts, the *non-statistical approach* of [117, 118] for analyzing the homology and the "genetic semi-homology" in protein sequences is another example of using knowledge about coding DNA when aligning amino acid sequences. Instead of using a statistically computed scoring matrix, amino acid similarities are scored according to the complexity of the substitution process at the DNA level, depending on the number and type (transition/transversion) of nucleotide changes that are necessary for replacing one amino acid by the other. This ensures a differentiated treatment of amino acid substitutions at different

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

positions of the protein sequence, thus avoiding possible rough approximations resulting from scoring them equally, based on a classic scoring matrix.

A more explicit use of the protein back-translation is given in [17]. The author proposed to search for protein homologies by aligning their *sequence graphs* with a variation of the algorithm for coding DNA alignment described in [16]. Basically, such a graph associated to a protein sequence is a directed acyclic graph with nodes labeled by nucleotides, where every maximal path describes a DNA sequence that can be translated into that protein. This is a straightforward compact representation of all the sequences that encode a protein, suggested by the fact that the codons encoding an amino acid generally differ by the nucleotide on their third position. The next chapter will provide a more rigorous discussion and a more compact representation of this data type, denoted from this point forward by the more illustrative term *back-translation graph*. Thus, the notion of *back-translation* is extended to refer to all the putative DNA sequences, encoded by a graph structure. This allows an exhaustive exploration and alignment with potential frameshifts of all putative DNA sequences encoding two proteins. Although the number of such sequences may be exponential with respect to the length of the protein, their alignment can be performed in polynomial (quadratic) time thanks to this compact representation.

4.3 Contributions

Most of the methods described in Section 4.2 date from over a decade ago and has enjoyed rather little visibility since. One can distinguish recurring approaches, such as building score matrices for frameshifted amino acid similarities, or using both the DNA and its corresponding protein sequence, via translation or back-translation, to improve similarity detection in the comparison of sequences affected by frameshifts. Although emerged and initially conducted independently from them, the algorithms proposed in the following chapters revive some ideas explored in the aforementioned efforts, bringing several scientific and technical variations and improvements, and eventually leading to a first (to our knowledge) fully featured, freely available implementation of a protein alignment approach targeting frameshift detection.

Basically, the goal of this work is to provide a method for *finding hidden protein homologies whose divergence is caused by a frameshift, by direct protein comparison*. While the same method can be used for any protein comparison task, it makes more sense in two particular contexts:

- When the proteins are not obtained from sequenced transcripts, but come directly from mass spectrometry, and there is no coding DNA available for making comparisons and detecting frameshifts at the DNA level.
- When the coding DNA is available, but the degree of divergence is so high that it cannot be handled by DNA comparison alone.

This approach can also detect less distant protein homologies, with or without frameshifts, or to obtain evidence of indel sequencing errors, but these use cases are not its primary goal.

The proposed method relies on the concept of *protein back-translation*. First, the complete set of DNA sequences encoding the proteins of interest is retrieved in the compact form as *back-translation graphs* (Section 5.1). Second, a new *dynamic programming algorithm for aligning two back-translation graphs* is proposed, which builds expressive alignments between the hypothetical nucleotide sequences obtained by back-translating the proteins, that can provide some information about the common ancestral sequence, if such a sequence exists. Frameshifts are handled by within the algorithm via a non-monotonic gap penalty function, where insertions and

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

deletions of full codons are less penalized than reading frame disruptive gaps. Additionally, since frameshifts are considered to be very rare events, the algorithm offers the possibility to restrict their number in an alignment by other means than very strong frameshift penalties which may bias the score statistics. This algorithm, described in Section 5.2, is simpler than its predecessors in that it handles pairs of symbols individually, and not as sequence segments. However, this simplification does not trigger the loss of contextual information, which is captured by a *powerful scoring system* designed to reflect the actual evolution process from a codon-oriented perspective, as explained in Section 5.3. Chapter 6 presents preliminary work on a heuristic seed-based approach for similarity search on all possible frameshifts. The proposed seeds basically take into account matches at the codon level, and allow the use of the same index for similarity search on different frameshifts. The implementation of this method is presented in Chapter 7, along with some experimental results and their discussion.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 5

Alignment of protein sequences with a frameshift

This chapter presents an algorithm for aligning protein sequences with a frameshift. The method does not rely on the actual coding DNA of each protein. Instead, the proteins are back-translated as graphs comprising all the putative DNA sequences in a compact form, as explained in Section 5.1, and a dynamic programming method for aligning such graphs is proposed in Section 5.2.

5.1 Back-translation graphs

The number of putative DNA sequences that can be translated into the same protein is, on average, exponential with respect to the length of the protein. This estimation follows from the number of codons encoding each of the amino acids that compose a protein. According to the standard genetic code, presented earlier in Chapter 1, Figure 1.12, the encoding of amino acids by codons is non-ambiguous and redundant, i.e. some amino acids are encoded by several different codons. In fact, with the exception of M and W, which have a single corresponding codon each, all amino acids are encoded by 2 (C, D, E, F, H, K, N, Q, Y), 3 (I), 4 (A, G, P, T, V) or 6 (L, R, S) codons. Assuming an amino acid composition similar to the figures given by Table 5.1, corresponding to the standard genetic code, one can expect about 95% of all amino acids composing a protein to be encoded by at least 2 codons, which yields an exponential explosion of coding sequence possibilities. Basically, a protein sequence of length n with the approximate amino acid composition of Table 5.1 is expected to have a number of putative coding DNA sequences in the order of 3^n.

Nevertheless, these coding sequences are very similar, sharing identical nucleotides on roughly more than half of their length on average. This results from the simple observation that, for any amino acid encoded by 2-4 codons, all these codons are identical on the first and second position, and differ only by their last. Similarly, when 6 different codons encode the same amino acid, they can be separated into two groups of 4 and 2 respectively that have the same property.

5.1.1 Back-translation graph representation

Based on these observations, all the putative DNA sequences for a given protein can be represented efficiently as a directed acyclic graph, with nodes labeled by nucleotide symbols, and paths describing putative DNA sequences. As illustrated in Figure 5.1 (a), the graph is basically organized in $3n$ positions, where n is the length of the protein sequence. Every position i of the

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Amino Acid	Percentage	Amino Acid	Percentage
A (Ala)	8.3%	M (Met)	2.4%
C (Cys)	1.7%	N (Asn)	4.4%
D (Asp)	5.3%	P (Pro)	5.1%
E (Glu)	6.2%	Q (Gln)	4.0%
F (Phe)	3.9%	R (Arg)	5.7%
G (Gly)	7.2%	S (Ser)	6.9%
H (His)	2.2%	T (Thr)	5.8%
I (Ile)	5.2%	V (Val)	6.6%
K (Lys)	5.7%	W (Trp)	1.3%
L (Leu)	9.0%	Y (Tyr)	3.2%

Table 5.1: Amino acid distribution according to the standard genetic code.

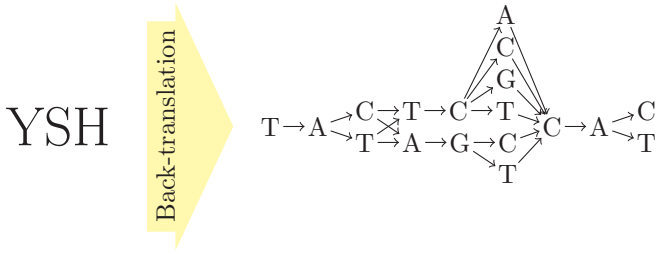

Figure 5.1: Back-translation graph example for the amino acid sequence YSH, describing all putative DNA sequences.

graph has one or more corresponding nodes, each node representing a possible nucleotide that can appear at position i in at least one of the putative coding sequences.

For identical nucleotides that appear at the same position of different codons for the same amino acid, and are preceded by different nucleotides within their respective codon, (as it is the case for bases C and T at the second position of the codons corresponding to amino acids S and L respectively), different nodes with the same label are introduced into the graph in order to avoid the creation of paths that do not correspond to actual putative DNA sequences for the given protein. Also, as the scoring system proposed in Section 5.3 requires to differentiate identical symbols by their context, identical nucleotides appearing at the second and third position of different codons for amino acids L, S and R will have different corresponding nodes in the back-translation graph. Basically, each nucleotide symbol α from a putative coding DNA sequence belonging to some codon c is *labeled with a word w which is its prefix in the codon c*. Depending on the position of α in c, w has length 0, 1 or 2. Here we denote such a labeled symbol by $\alpha^{(w)}$.

For example, let us consider the 6 codons encoding the amino acid R: AGA, AGG, CGA, CGC, CGG, CGT. All nucleotides on the first positions of these codons have empty prefixes within their codons, and are therefore labeled with the empty word ϵ: $A^{(\epsilon)}$, $A^{(\epsilon)}$, $C^{(\epsilon)}$, $C^{(\epsilon)}$, $C^{(\epsilon)}$, $C^{(\epsilon)}$ respectively. Thus, there are 2 unique labeled symbols, $A^{(\epsilon)}$ appearing twice and $C^{(\epsilon)}$ appearing four times. On the second position of each codon we find the nucleotide G, but with different prefixes: $G^{(A)}$ in the first two codons and $G^{(C)}$ in the last four. Finally, the labeled symbols on the last position of the codons are $A^{(AG)}$, $G^{(AG)}$, $A^{(CG)}$, $C^{(CG)}$, $G^{(CG)}$, $T^{(CG)}$. Here there are two nucleotides (A and G) appearing on the third position of two different codons each, but with different prefixes (or labels w), namely AG and CG.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

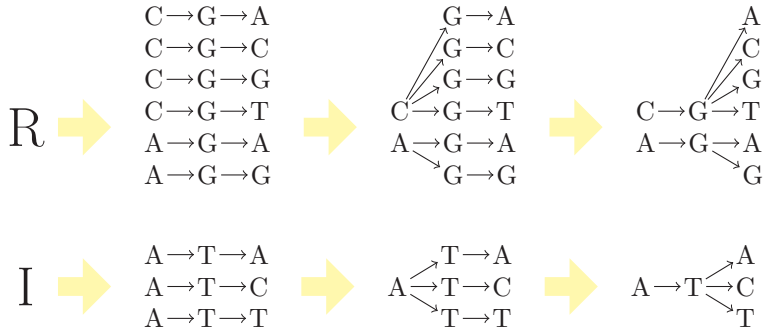

Figure 5.2: The construction of simple back-translation graphs for the amino acids R, encoded by 6 codons, and I, encoded by 3 codons. Identical nucleotides are associated to different nodes if they have different prefixes in the codons where they appear.

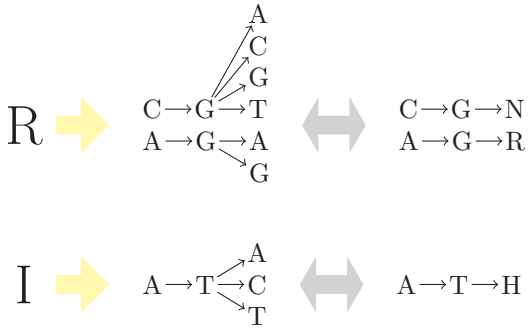

Figure 5.3: Back-translation graphs using ambiguous nucleotide codes for the amino acids R, encoded by 6 codons, and I, encoded by 3 codons.

In later formalizations the w label may be dropped for notation simplicity, and this differentiation will be considered implicit. Two symbols that appear at the same position of two putative DNA sequences encoding the same protein are identical (and are thereby represented by the same node) if and only if they represent the same nucleotide and their labels are identical. Two nodes at consecutive positions are linked by an arc if and only if they are either consecutive nucleotides of the same codon, or they are respectively the third and the first base of two consecutive codons. No other arcs exist in the graph. The construction of a simple back-translation graph for the amino acids R and I is illustrated in Figure 5.2.

5.1.2 Compact back-translation graph representation using IUPAC ambiguity codes

The graph representing the set of putative DNA sequences for a protein has a number of nodes and arcs that depend linearly on the length n of the protein sequence. This follows directly from the fact that the graph models $3n$ positions in a DNA sequence, with no more than 6 nodes each, and there are arcs only between nodes on consecutive positions. The graph size can be further decreased if we represent all the different symbols $\alpha_i^{(w)}$ appearing at the same position i of the graph and having identical non-empty labels (prefixes within their codon) w by a single node whose name is given by the IUPAC ambiguity code [42] denoting the set $\{\alpha_i\}$. These nucleotide ambiguity codes (given previously in Chapter 3, Table 3.1) enable the representation of all the amino acids encoded by 2-4 codons by a single sequence of 3 symbols, where the 3rd

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

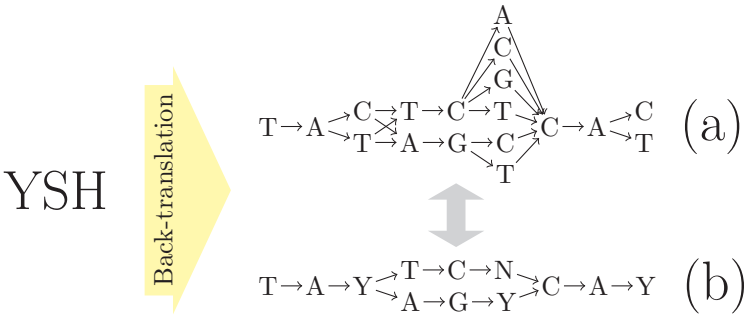

Figure 5.4: Back-translation graph example for the amino acid sequence YSH: (a) explicitly describing all putative DNA sequences and (b) compact representation using ambiguity codes.

symbol may account for 1, 2, 3 or 4 nucleotides. The amino acids encoded by 6 codons can be represented by two such segments of 3 symbols, as shown in Table 5.1.2. Considering this set of "ambiguous" codons for each amino acid, a back-translation graph can be constructed in the similar manner as described above, as illustrated in Figure 5.3 and Figure 5.4. Under this nucleotide encoding, the size of the back-translation graph is dramatically reduced, since most amino acid (78% according to Table 5.1) are expected to be back-translated into a single path of length 3, and the others require 6 nodes (two paths of length 3) to be represented.

A more formal definition of the back-translation graph is given below. The ambiguity codes are interchangeably used as symbols denoting a node in the graph, and sets of nucleotides they represent. For instance, $R \equiv \{A, G\}$, $N \equiv \{A, C, G, T\}$, $C \equiv \{C\}$. The definition stands whether the ambiguity codes are used or not, as long as it is accepted that $\alpha \equiv \{\alpha\}, \forall \alpha \in \{A, C, G, T\}$.

Definition 11. *A* back-translation graph *associated to a protein sequence P of length n is a directed acyclic graph $G_P = (V_P, E_P)$ where:*

$$V_P = \bigcup_{i=1}^{3 \cdot n} \{\alpha_i^{(w)} | \forall x \in \alpha_i, \exists D \in \{A, C, G, T\}^{3n} : P = translation(D) \wedge w \cdot x \in suffix(D[1..i])\} \tag{5.1}$$

where $\alpha_i^{(w)}$ is a symbol that appears at position i in the graph, denoting a set of nucleotides that appear on position i of at least one of the protein's putative coding sequences and has the prefix w in its codon (as explained earlier in Section 5.1.1), and

$$E_P = \{(\alpha_i^{(w_1)}, \alpha_{i+1}^{(w_2)}) \quad | \quad \begin{array}{l} \exists D \in \{A, C, G, T\}^{3n} : P = translation(D) \wedge \\ \exists x \in \alpha_i : w_1 \cdot x \in suffix(D[1..i]) \wedge \\ \exists x' \in \alpha_{i+1} : w_2 \cdot x' \in suffix(D[1..i+1])\} \end{array} \tag{5.2}$$

are arcs between nodes corresponding to symbols that are consecutive in one of the protein's putative coding sequences.

A similar definition was given in [17], with the remark that it explicitly enumerated all the nucleotide possibilities on the third position of the codon, basically because the associated alignment algorithm was not designed to work with ambiguity codes. On the other hand, the scoring system proposed here in Section 5.3 allows the use of this compact form, with no information loss.

In the remainder of this chapter, the term *back-translation graph* will refer to the compact representation using ambiguity codes, and *expanded back-translation graph* will denote back-translation graphs where possible nucleotides on each position are explicitly represented by

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Amino Acid	Codons	Amino Acid	Codons
A (Ala)	GC**N**	M (Met)	ATG
C (Cys)	TG**Y**	N (Asn)	AA**Y**
D (Asp)	GA**Y**	P (Pro)	CC**N**
E (Glu)	GA**R**	Q (Gln)	CA**R**
F (Phe)	TT**Y**	R (Arg)	AG**R**, CG**N**
G (Gly)	GG**N**	S (Ser)	AG**Y**, TC**N**
H (His)	CA**Y**	T (Thr)	AC**N**
I (Ile)	AT**H**	V (Val)	GT**N**
K (Lys)	AA**R**	W (Trp)	TGG
L (Leu)	TT**R**, CT**N**	Y (Tyr)	TA**Y**

Table 5.2: Sets of codons encoding each amino acid, represented using the IUPAC ambiguity codes.

nodes in the graph. The latter is occasionally used in this chapter in examples only. The implementation of this method is based on the compact representation, which helps to save both time and memory.

5.1.3 Space complexity

The space necessary for storing the back-translation graph of a protein sequence P of size n depends linearly on n. Basically, as mentioned in Section 5.1.2, the back-translation graph $G_P = (V_P, E_P)$ consists of $3 \cdot n$ groups of nodes $\{\alpha_i^{(w)}\}$ (as each of the n amino-acids are encoded by sequences of 3 nucleotides). In the expanded form of the back-translation graph, every group i contains the nodes corresponding to the nucleotides that appear at position i in at least one of the putative coding sequences. The number of nodes in a group is limited by the number of codons that encode an amino acid, which is 6 in the worst case scenario for non-ambiguous symbols, thus does not depend on the protein's length. Arcs exist only between nodes in consecutive groups (equation (5.2)), therefore each node can have a limited number of neighbors, also not depending on the protein's length. Consequently, the overall memory consumption for storing the expanded back-translation graph of a protein sequence P of size n is $\mathcal{O}(n)$, with a number of nodes necessary for the representation of each amino acids ranging between 3 and 10. The worst case scenario is a protein sequence composed only of the amino acids L, S, R, which are encoded by 6 codons each, and hence have the most complex back-translation. For each such amino acid, 10 nodes and 20 arcs are necessary (as depicted in Figure 5.2), yielding a maximum memory size of $30n$ for the entire graph.

The reasoning is similar for the compact back-translation graphs, where the ambiguous nucleotide symbol encoding is used. However, in this representation, only 6 nodes and 6 arcs are necessary in the worst case for an amino acid, while most amino acids require just 3 nodes and 3 arcs for their back-translation.

5.1.4 Reverse complementary of a back-translation graph

The reverse complementary of a back-translation graph can be obtained in a classic manner, by reversing the arcs and complementing the nucleotide symbols that label the nodes, as illustrated in Figure 5.5. The complementary of an ambiguous nucleotide symbol α is the symbol denoting the set of nucleotides that are complementary to the ones contained in the set represented by

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

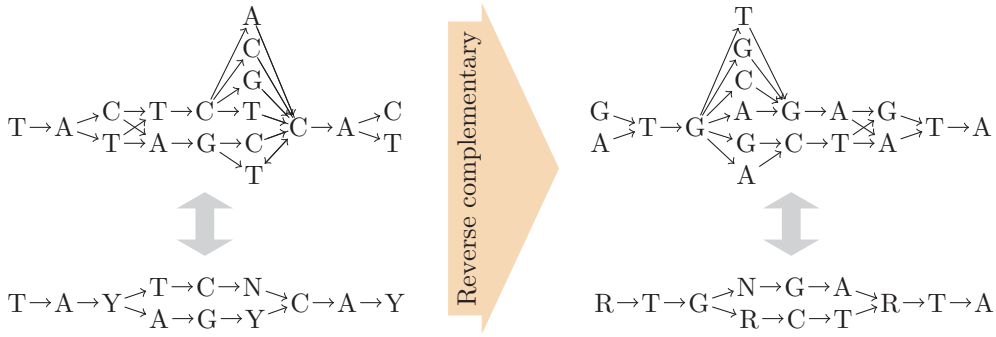

Figure 5.5: Example of reverse complementary back-translation graphs for the amino acid sequence YSH: The reverse complementary of a back-translation graph can be obtained in a classic manner, by reversing the arcs and complementing the nucleotide symbols that label the nodes.

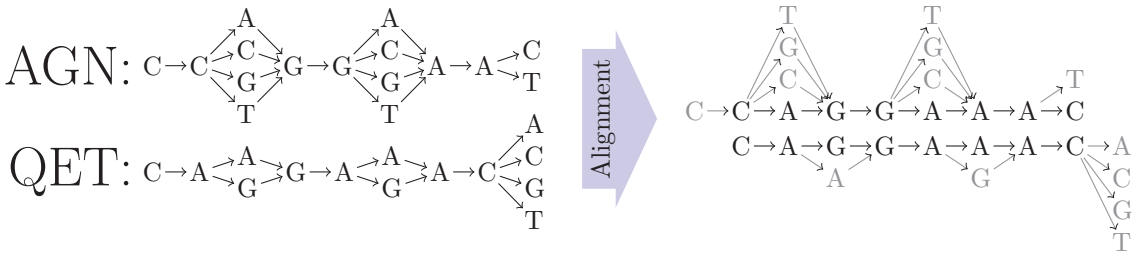

Figure 5.6: Alignment example: A path (corresponding to a putative DNA sequence) was chosen from each graph so that the match/mismatch ratio is maximized.

α. Thus, the complementaries of R, Y, H and N are Y, R, D and N respectively.

5.2 Alignment algorithm

When aligning two back-translation protein sequences, the goal is to find the two putative DNA sequences (one for each protein) that are most similar with respect to an established scoring scheme, as shown in a small example in Figure 5.6. By extending Definition 1, an alignment of two back-translation graphs is defined as follows:

Definition 12. *Given two back-translation graphs G_A and G_B, their alignment consists of two equally sized sequences $S'_A = a_1...a_l$ and $S'_B = b_1...b_l$, over an alphabet $\Sigma \cup \{-\}$ (where Σ is the alphabet of nucleotides, and $- \notin \Sigma$ represents a gap in the alignment) obtained by inserting zero or more gaps between the symbols of two DNA sequences S_A and S_B comprised in the graphs G_A and G_B respectively, in order to shift on the same position intervals subsequences that are similar in both S_A and S_B, with the constraint that $\nexists h \in \{1, ..., l\} : a_h = b_h = -$.*

This section presents a dynamic programming method, inspired from the Smith-Waterman algorithm [188], designed to align back-translation graphs and handle frameshifts via its gap related restrictions.

5.2.1 The dynamic programming table

Given the input graphs G_A and G_B obtained by back-translating proteins P_A and P_B, the algorithm finds the best scoring local alignment between two DNA sequences comprised in the

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

back-translation graphs. Partial solutions are stored in a table M. For each entry $M[i, j, (\alpha_i, \beta_j)]$, i and j are positions of G_A and G_B respectively, and (α_i, β_j) enumerates the possible pairs of nodes that can be found in G_A at position i, and in G_B at position j, respectively. In consequence, $M[i, j, (\alpha_i, \beta_j)]$ holds the score of the best alignment *ending at positions i and j respectively* between two prefixes $D^A_{1..i}$ and $D^B_{1..j}$ of the DNA sequences D^A and D^B with $translation(D^A) = P_A$ and $translation(D^B) = P_B$, such that $D^A_i \in \alpha_i$ and $D^B_j \in \beta_j$.

An example of table M is given in Figure 5.7 for expanded back-translation graphs, and Figure 5.8 depicts the table M for the equivalent condensed back-translation graphs. Note that the back-translation graphs are 1-indexed, while the table M is 0-indexed, i.e. $M[0, 0, \cdot]$, $M[i, 0, \cdot]$, $M[0, j, \cdot]$ are defined. This apparent inconsistency is necessary in order to allow the storage in M of partial solutions (scores) for alignments consisting of gaps being inserted *before* one of the sequences, which are necessary in the initialization step (relation (5.3)), and at the same time to ensure that the definition of $M[i, j, (\alpha_i, \beta_j)]$ given above holds under this condition.

5.2.2 Recurrence relations

The dynamic programming algorithm begins with a classic local alignment **initialization** of M, i.e. 0 at the top and left borders, as shown in (5.3), where the symbol '-' replacing a graph node denotes the fact that only gaps have been inserted before one of the sequences in the corresponding partial alignment.

$$\begin{aligned} M[0, 0, (-, -)] &= 0 \\ M[i, 0, (\alpha_i, -)] &= 0 \\ M[0, j, (-, \beta_j)] &= 0 \end{aligned} \tag{5.3}$$

The **recursion** step is described by relation (5.5). The partial alignment score of each entry $M[i, j, (\alpha_i, \beta_j)]$ is computed as the maximum of 6 types of values:

(a) 0 (similarly to the classic Smith-Waterman algorithm, only positive scores are considered for local alignments).

(b) the substitution score of symbols (α_i, β_j), denoted $score(\alpha_i, \beta_j)$, added to the score of the best partial alignment ending in $M[i-1, j-1]$, provided that the partially aligned paths contain α_i at position i and β_j on position j respectively; this condition is ensured by restricting the entries of $M[i-1, j-1]$ to those labeled with symbols that precede α_i and β_j in the graphs, and is expressed in (5.5) by $\alpha_{i-1} \in pred_{G_A}(\alpha_i)$, $\beta_{j-1} \in pred_{G_B}(\beta_j)$.

(c) the cost $fsGapPenalty$ of a deletion frameshift (gap of size 1 or extension of a gap of size 1) in sequence A, added to the score of the best partial alignment that ends in a cell $M[i, j-1, (\alpha_i, \beta_{j-1})]$, provided that β_{j-1} precedes β_j in the second graph ($\beta_{j-1} \in pred_{G_B}(\beta_j)$).

(c') the cost $fsExtensionPenalty$ of a deletion frameshift (extension of a gap of size 1) in sequence A, added to the score of the best partial alignment that ends in a cell $M[i, j-1, (\alpha_i, \beta_{j-1})]$, provided that β_{j-1} precedes β_j in the second graph ($\beta_{j-1} \in pred_{G_B}(\beta_j)$); this case is considered only if the previous pair on the current partial alignment is a frameshift gap in sequence A.

(d) the cost of a deletion frameshift in sequence B, added to a partial alignment score defined as in (c).

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

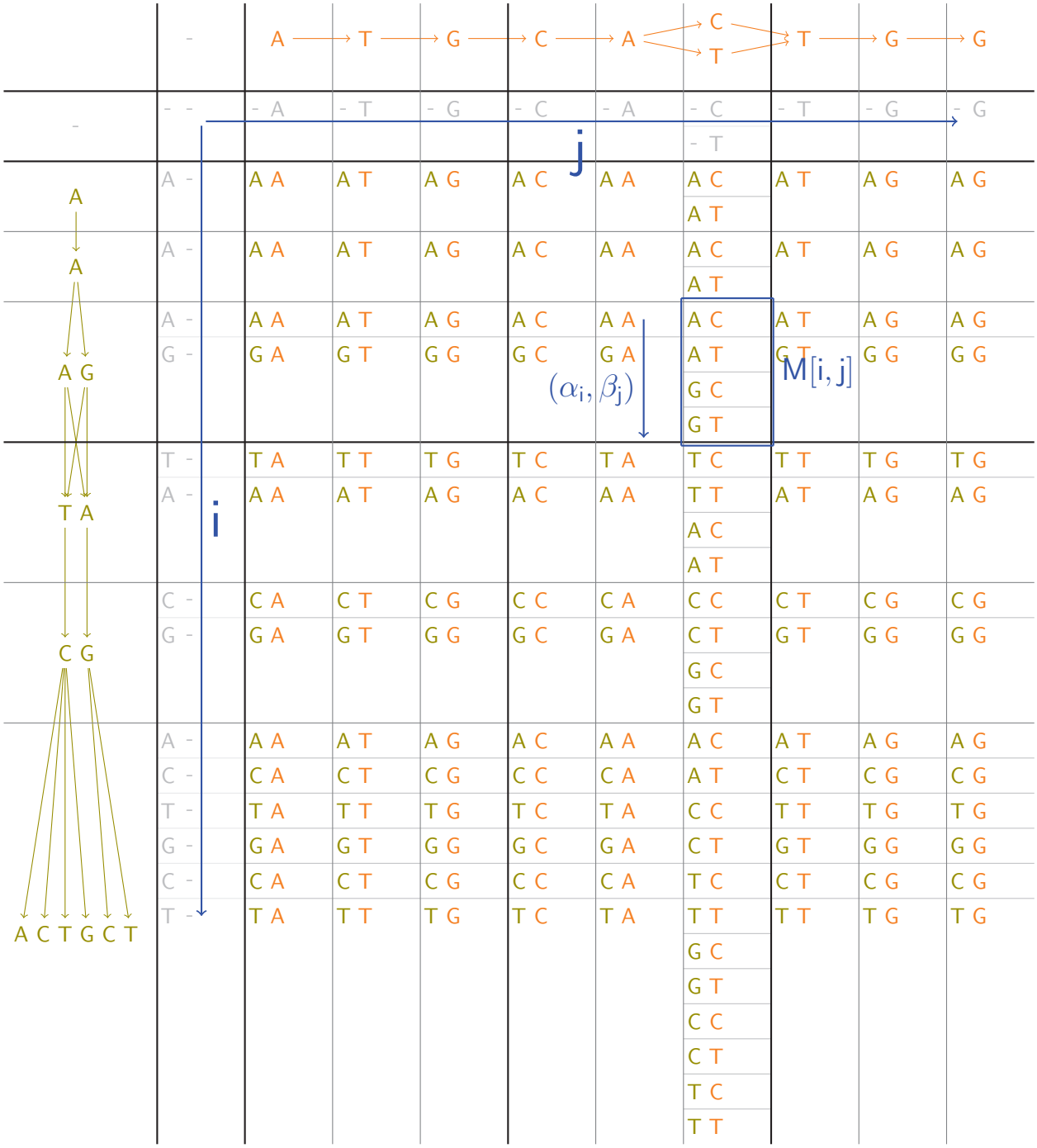

Figure 5.7: Example of dynamic programming table M for computing the best local alignment of two expanded back-translation graphs: $M[i, j]$ is a "cell" of M corresponding to position i of the first graph and position j of the second graph, and contains entries (α_i, β_j) corresponding to pairs of nodes occurring in the first graph at position i, and in the second graph at position j, respectively.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

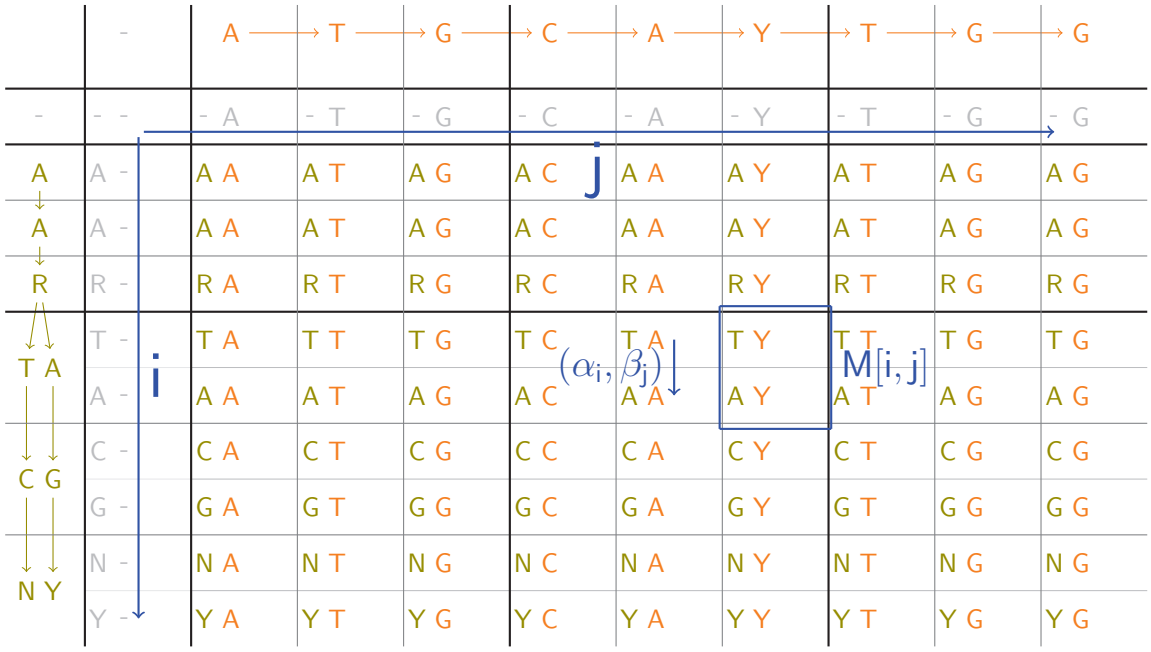

Figure 5.8: Example of dynamic programming table M for computing the best local alignment of two compact back-translation graphs: similarly to Figure 5.7, $M[i, j]$ is a "cell" of M corresponding to position i of the first graph and position j of the second graph, and contains entries (α_i, β_j) corresponding to pairs of nodes occurring in the first graph at position i, and in the second graph at position j, respectively.

(d') the cost $fsExtensionGapPenalty$ of a deletion frameshift (extension of a gap of size 1) in sequence B, added to a partial alignment score defined as in (c').

(e) the cost $tripleGapPenalty$ of skipping an entire codon from sequence B (i.e. an actual amino acid from P_B), added to the score of the best partial alignment ending in a cell $M[i, j-3, (\alpha_i, \beta_{j-3})]$, with no restrictions imposed to β_{j-3}.

(f) the cost of skipping an entire codon from sequence A, added to the score of the best partial alignment ending in a cell $M[i-3, j, (\alpha_{i-3}, \beta_j)]$, also with no restrictions on α_{i-3}. In practice, the codon gap penalty is smaller than the cumulated penalty for 3 frameshifts, as explained in Section 5.2.3.

The recurrence relations are illustrated in Figure 5.9 and Figure 5.10. When aligning expanded back-translation graphs (Figure 5.9), for indices i and j corresponding to the last position of the codon, the number of choices can be significantly larger than in the case of compact back-translation graphs (Figure 5.10), since in the former all nucleotides on the third codon position are explicitly represented by nodes. Nevertheless, the scoring scheme ensures that relation (5.5) produces the same result in both cases. Namely, as will be shown in detail in Section 5.3, the score $\sigma(\alpha, \beta)$ of two ambiguous symbols α and β is pre-computed as

$$\sigma(\alpha, \beta) = \max_{x \in \alpha, y \in \beta} \sigma(x, y). \tag{5.4}$$

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

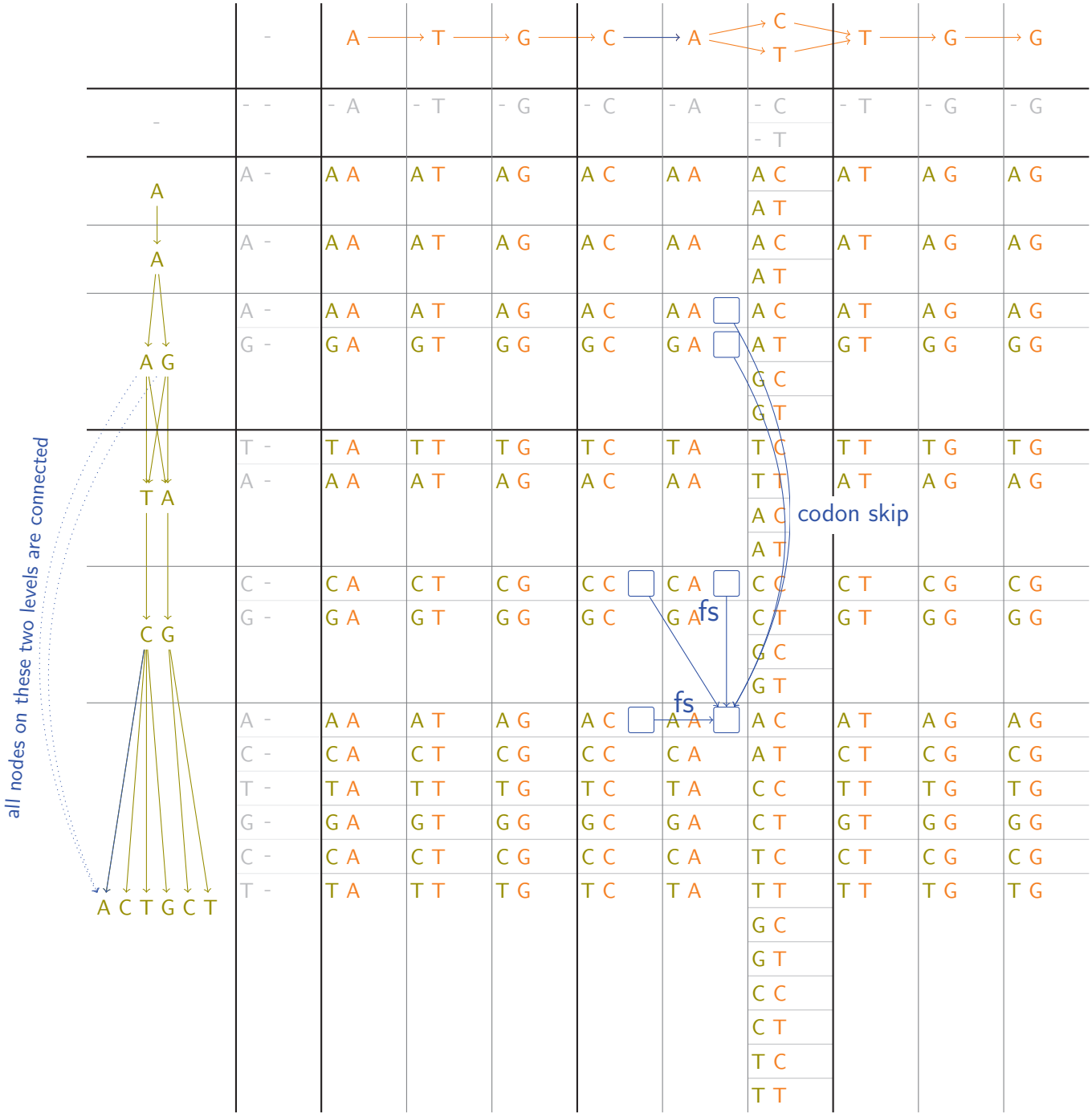

Figure 5.9: Illustration of the recurrence relations given by (5.5) for computing partial solutions in a dynamic programming table that will give the best scoring alignment between two expanded back-translation graphs.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

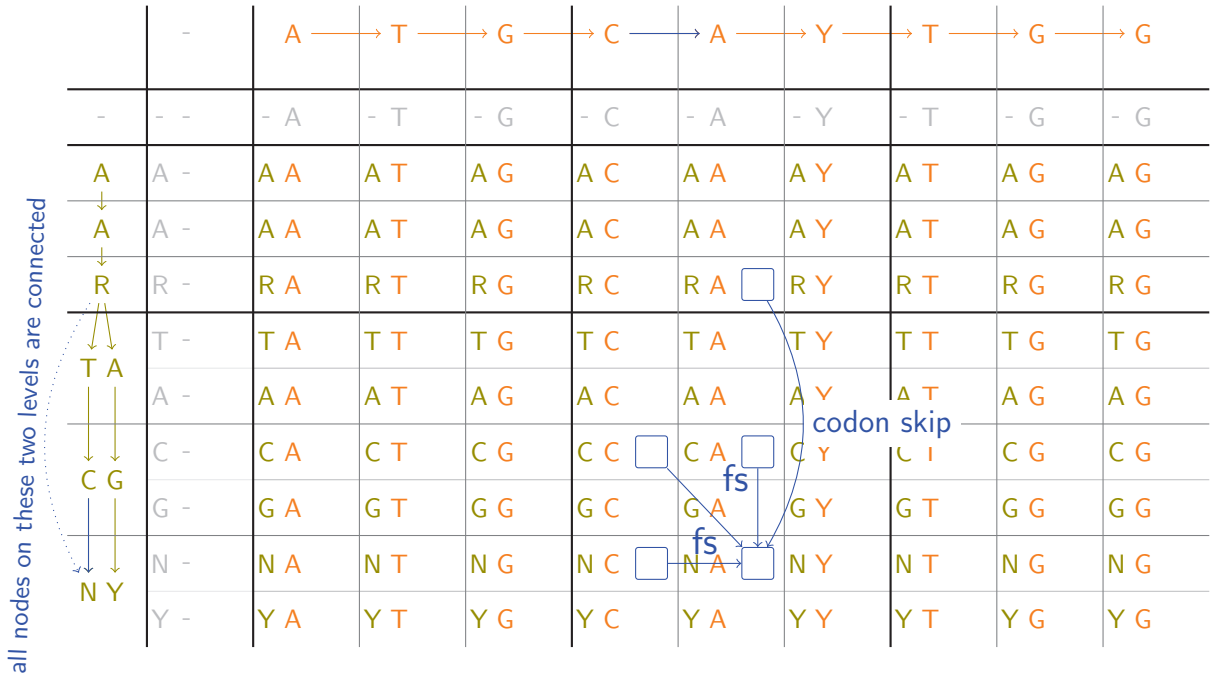

Figure 5.10: Illustration of the recurrence relations given by (5.5) for computing partial solutions in a dynamic programming table that will give the best scoring alignment between two compact back-translation graphs. The number of table entries to interrogate for computing a new partial solution in generally smaller than in the case of aligning the equivalent expanded back-translation graphs, depicted in Figure 5.9.

$$
M[i,j,(\alpha_i,\beta_j)] = \begin{cases} 0 & & \text{(a)} \\ \max \begin{cases} M[i-1,j-1,(\alpha_{i-1},\beta_{i-1})] + \sigma(\alpha_i,\beta_j), & \alpha_{i-1} \in pred_{G_A}(\alpha_i); \ \beta_{j-1} \in pred_{G_B}(\beta_j); & \text{(b)} \\ M[i,j-1,(\alpha_i,\beta_{j-1})] + fsGapPenalty, & \beta_{j-1} \in pred_{G_B}(\beta_j); & \text{(c)} \\ M[i,j-1,(\alpha_i,\beta_{j-1})] + fsExtensionPenalty, & \beta_{j-1} \in pred_{G_B}(\beta_j); & \text{(c')} \\ \quad \textit{(only if the previous entry on the alignment path is in } M[i,j-2]) & & \\ M[i-1,j,(\alpha_{i-1},\beta_j)] + fsGapPenalty, & \alpha_{i-1} \in pred_{G_A}(\alpha_i); & \text{(d)} \\ M[i-1,j,(\alpha_{i-1},\beta_j)] + fsExtensionPenalty, & \alpha_{i-1} \in pred_{G_A}(\alpha_i); & \text{(d')} \\ \quad \textit{(only if the previous entry on the alignment path is in } M[i-2,j]) & & \\ M[i,j-3,(\alpha_i,\beta_{j-3})] + tripleGapPenalty, & j \geq 3, j \text{ multiple of 3}; & \text{(e)} \\ M[i-3,j,(\alpha_{i-3},\beta_j)] + tripleGapPenalty, & i \geq 3, i \text{ multiple of 3};. & \text{(f)} \end{cases} \end{cases} \tag{5.5}
$$

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

5.2.3 Handling frameshifts

The algorithm adopts a non-monotonic gap penalty function, where insertions and deletions of full codons are penalized less than reading frame disruptive gaps. More precisely, as can be seen in equation (5.5), two particular kinds of gaps are considered: **frameshifts** – gaps of size 1 or 2, with high penalty, and **codon skips** – gaps of size 3 which correspond to the insertion or deletion of a whole codon. The fact that the penalty for frameshifts is higher than the penalty for in-frame gaps ensures that no gap of length g multiple of 3 is wrongfully considered as a series of frameshifts.

Additionally, since frameshifts are considered to be very rare events, one may be interested in imposing a maximum number F of frameshifts in the alignments. To cope with this restriction, the alignment algorithm described above is adapted in order to find alignments that have at most F frameshifts as explained below.

Let G_A and G_B be the back-translation graphs to align, obtained by back-translating proteins P_A and P_B. We maintain $F+1$ dynamic programming tables M^f, $\forall f \in \{0, ..., F\}$, organized as explained in Section 5.2.1. The entry $M^f[i, j, (\alpha_i, \beta_j)]$ now holds the score of the best alignment **with f frameshifts** between two prefixes $D^A_{1..i}$ and $D^B_{1..j}$ of the DNA sequences D^A and D^B with $translation(D^A) = P_A$ and $translation(D^B) = P_B$, such that $D^A_i \in \alpha_i$ and $D^B_j \in \beta_j$.

The **initialization** of M^f is done according to relations (5.6) – (5.8). This initialization differs slightly from (5.3), and basically ensures that the correct number of frameshifts is taken into account for alignments starting on the top-left border of M. As such, alignments beginning at positions (i, j) with $i = 0$ or $j = 0$ with $i - j$ multiple of 3 start "in-frame" and should appear in M^0, while those where $i - j$ is not multiple of 3 start with a frameshift and belong in M^1. All other entries on the top-left borders of M^f are invalidated by setting their value to $-\infty$, which forces the algorithm to discard them when recursively computing the partial solutions. Note that, in alignments starting with a reading frame difference, the initial frameshift is counted but not penalized.

$$\begin{aligned}
M^0[0,0,(-,-)] &= 0 \\
M^0[i,0,(\alpha_i,-)] &= \begin{cases} 0, & i \text{ multiple of 3;} \\ -\infty, & i \text{ not multiple of 3.} \end{cases} \\
M^0[0,j,(-,\beta_j)] &= \begin{cases} 0, & j \text{ multiple of 3;} \\ -\infty, & j \text{ not multiple of 3.} \end{cases}
\end{aligned} \tag{5.6}$$

$$\begin{aligned}
M^1[0,0,(-,-)] &= -\infty \\
M^1[i,0,(\alpha_i,-)] &= \begin{cases} -\infty, & i \text{ multiple of 3;} \\ 0, & i \text{ not multiple of 3.} \end{cases} \\
M^1[0,j,(-,\beta_j)] &= \begin{cases} -\infty, & j \text{ multiple of 3;} \\ 0, & j \text{ not multiple of 3.} \end{cases}
\end{aligned} \tag{5.7}$$

$$\begin{aligned}
M^f[0,0,(-,-)] &= -\infty, \forall f \in \{2, ..., F\} \\
M^f[i,0,(\alpha_i,-)] &= -\infty, \forall f \in \{2, ..., F\} \\
M^f[0,j,(-,\beta_j)] &= -\infty, \forall f \in \{2, ..., F\}.
\end{aligned} \tag{5.8}$$

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

The **recursion** step is described by relation (5.9). The main difference from (5.5) appears in options (c) and (d), which state that opening a frameshift gap creates a "jump" of the alignment path from M^{f-1} to M^f, thus taking into account the correct number f of frameshifts along the current alignment path. Note that options (c) and (d) are disabled for $M(0)$. Also, since an alignment cannot start with more than one frameshift, the option 0 **(a)** of relation (5.9), which marks the beginning of a potential local alignment, is valid only for the entries $M^f[i, j, (\alpha_i, \beta_j)]$ where: i) either $f = 0$ and $i - j$ is multiple of 3, meaning that the local alignment begins with both sequences on the same reading frame, or ii) $f = 1$ and $i - j$ is not multiple of 3, meaning that the local alignment begins with a reading frame difference (see also the initialization step and relations (5.6) – (5.8)). This prevents from initializing with a wrong value the number of frameshifts for local alignments that begin at positions (i, j) with $i > 0$ and $j > 0$.

$$
M^f[i,j,(\alpha_i,\beta_j)] = \max \begin{cases}
0 & \text{if } f=0 \text{ and } (i-j) \text{ multiple of 3} \\ & \text{or } f=1 \text{ and } (i-j) \text{ not multiple of 3;} \quad (\mathbf{a}) \\
M^f[i-1,j-1,(\alpha_{i-1},\beta_{i-1})] + \sigma(\alpha_i,\beta_j), & \alpha_{i-1} \in pred_{G_A}(\alpha_i); \\ & \beta_{j-1} \in pred_{G_B}(\beta_j); \quad (\mathrm{b}) \\
M^{f-1}[i,j-1,(\alpha_i,\beta_{j-1})] + fsGapPenalty, & \beta_{j-1} \in pred_{G_B}(\beta_j), f>0; \quad (\mathbf{c}) \\
M^f[i,j-1,(\alpha_i,\beta_{j-1})] + fsExtensionPenalty, & \beta_{j-1} \in pred_{G_B}(\beta_j), f>0; \quad (\mathrm{c}') \\
\textit{(if the previous entry on the alignment path is in } M^{f-1}[i,j-2]) & \\
M^{f-1}[i-1,j,(\alpha_{i-1},\beta_j)] + fsGapPenalty, & \alpha_{i-1} \in pred_{G_A}(\alpha_i), f>0; \quad (\mathbf{d}) \\
M^f[i-1,j,(\alpha_{i-1},\beta_j)] + fsExtensionPenalty, & \alpha_{i-1} \in pred_{G_A}(\alpha_i), f>0; \quad (\mathrm{d}') \\
\textit{(if the previous entry on the alignment path is in } M^{f-1}[i-2,j]) & \\
M^f[i,j-3,(\alpha_i,\beta_{j-3})] + tripleGapPenalty, & j \geq 3, j \text{ multiple of 3;} \quad (\mathrm{e}) \\
M^f[i-3,j,(\alpha_{i-3},\beta_j)] + tripleGapPenalty, & i \geq 3, i \text{ multiple of 3;}. \quad (\mathrm{f})
\end{cases}
\tag{5.9}
$$

The score of the best alignment between the back-translation graphs G_A and G_B is given by the entry $M^{f_b}[i_b, j_b, (\alpha^b_{i_b}, \beta^b_{j_b})] = max_{f,i,j,\alpha_i,\alpha_j} M^f[i, j, (\alpha_i, \beta_j)]$ holding the *best score*, i.e. the highest value in the dynamic programming table.

5.2.4 Traceback: retrieving the alignment

Let $C^f[i, j, (\alpha_i, \beta_j)] = [f', i', j', (\alpha'_{i'}, \beta'_{j'})]$ denote the traceback coordinates of the entry $M^{f'}[i', j', (\alpha'_{i'}, \beta'_{j'})]$ whose value is chosen in the recurrence relation (5.9) for computing $M^f[i, j, (\alpha_i, \beta_j)]$. Note that, if there is no restriction on the number of frameshifts allowed in an alignment, then the algorithm given by (5.5) is used instead of (5.9), and the value f is therefore ignored. The best alignment can be traced back from $M^{f_b}[i_b, j_b, (\alpha^b_{i_b}, \beta^b_{j_b})]$, as indicated by this table C of traceback coordinates, until the first entry $M^{f_s}[i_s, j_s, (\alpha^s_{i_s}, \beta^s_{j_s})] = 0$ is reached. However, if α and β are ambiguous symbols, this procedure does not retrieve the actual DNA sequences D^b_A and D^b_B yielding the highest alignment score. To obtain them, an additional

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

matrix τ is pre-computed together with the scoring matrix σ, described in Section 5.3, storing the actual pair of non-ambiguous nucleotide symbols $x \in \alpha$ and $y \in \beta$ that have the highest score and give the value of $\sigma(\alpha, \beta)$:

$$\tau(\alpha, \beta) = arg \max_{x \in \alpha, y \in \beta} \sigma(x, y). \tag{5.10}$$

For reasons of consistency between concepts involving expanded and compact back-translation graphs, τ is defined in theory for alignments of expanded back-translation graphs as

$$\tau(\alpha, \beta) = (\alpha, \beta). \tag{5.11}$$

The traceback procedure is described by Algorithm 2, an extension of Algorithm 1 of Chapter 3, Section 3.2.3, adapted to local alignments of back-translation graphs. Starting from the cell $M^{f_b}[i_b, j_b, (\alpha^b_{i_b}, \beta^b_{j_b})]$ that holds the highest score, it builds two strings $D'_A = a'_1...a'_l$ and $D'_B = b'_1...b'_l$ of equal lengths l that describe the alignment of two putative DNA sequences $D_A = a_{i_1}...a_{i_k}$ and $D_B = b_{j_1}...b_{j_h}$ comprised in the back-translation graphs G_A and G_B respectively. D'_A and D'_B are basically the sequences D_A and D_B with gaps $(-)$ inserted between their symbols.

5.2.5 Complexity

Let G_A and G_B be graphs obtained by back-translating proteins P_A and P_B, of lengths n_A and n_B respectively. It was shown earlier in Section 5.1.3 that their sizes depend linearly on the back-translated protein lengths. The dynamic programming table M computed by the alignment algorithm will have $3 \cdot n_A + 1$ rows and $3 \cdot n_B + 1$ columns. Each cell $M[i, j]$ has several entries corresponding to the possible pairs of nodes from each sequence. The number of entries is bounded by the square of the number of nodes $\mathcal{C}$ that can appear on each position in the graph ($\mathcal{C}^2$). Consequently, the total number of entries in the table is at most $\mathcal{C}^2 \cdot (3 \cdot n_A + 1) \cdot (3 \cdot n_B + 1)$, hence $\mathcal{O}(n_A \cdot n_B)$. In addition to the table M, a table C *with the same number of entries as* M is used by the algorithm for storing the coordinates necessary for tracing back the alignment.

If there is a restriction F of the number of frameshifts allowed in an alignment, then $F + 1$ matrices M^f and C^f of size $\mathcal{O}(n_A \cdot n_B)$ are used, as explained in Section 5.2.3. In consequence, the total size of the dynamic programming tables is $\mathcal{O}((F + 1) \cdot n_A \cdot n_B)$. Nevertheless, this restriction makes sense only for small values of F (usually 1 or 2), which means a reasonable practical increase of used memory.

For computing each score in the dynamic programming table, the expressions that need to be evaluated are given by equation (5.5) (or (5.9)), by querying some of the entries from 7 other cells in the table. Since the number of entries in each cell is bounded by $\mathcal{C}^2$, this operation is considered to be performed in constant time. Consequently, the overall time complexity of the algorithm is $\mathcal{O}(n_A \cdot n_B)$, or $\mathcal{O}((F + 1) \cdot n_A \cdot n_B)$ with frameshift gap restrictions. To recover the best alignment and the two actual sequences that produce it, we use a traceback algorithm with an execution time depending linearly on the alignment length, which by definition cannot be larger than $(3 \cdot n_A + 3 \cdot n_B + 1)$.

5.3 Scoring scheme

This section introduces a new translation-dependent scoring system suitable for the alignment algorithm presented in Section 5.2. This scoring scheme incorporates information about possible

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Input:

Two back-translation graphs G_A and G_B;

The dynamic programming table M obtained by the alignment algorithm given by (5.6) – (5.9), holding the best-scoring alignment of G_A and G_B;

The table C of traceback coordinates.

Output: The best scoring alignment in the form of two strings D'_A and D'_B.

begin

`/* Start from the coordinates of the entry of` M `holding the` **highest score** `*/`

$[f, i, j, (\alpha_i, \beta_j)] \leftarrow [f_b, i_b, j_b, (\alpha^b_{i_b}, \beta^b_{j_b})]$;

$D'_A \leftarrow$ "" `// initialize with empty string`

$D'_B \leftarrow$ "" `// initialize with empty string`

while $M^f[i, j, (\alpha_i, \beta_j)] \neq 0$ **do**

$(f', i', j', (\alpha_{i'}, \beta_{j'})) = C^f[i, j, (\alpha_i, \beta_j)]$;

if $i' == i - 1$ *and* $j' == j - 1$ **then**

`// match or substitution:`

$(a, b) \leftarrow \tau(\alpha_i, \beta_j)$ `// see equation` (5.10)

$D_A \leftarrow D_A \cdot a$;

$D_B \leftarrow D_B \cdot b$;

else

if $i' == i$ **then**

`// gap in` D'_A`:`

$D_A \leftarrow D_A \cdot$ ' $-$ '$^{(j-j')}$;

$D_B \leftarrow D_B \cdot \langle$any DNA sequence comprised in a path in G_B starting at j' and ending at $j\rangle$;

else

`// gap in` D'_B`:`

$D_B \leftarrow D_B \cdot$ ' $-$ '$^{(i-i')}$;

$D_A \leftarrow D_A \cdot \langle$any DNA sequence comprised in a path in G_A starting at i' and ending at $i\rangle$;

end

end

`/* Go to the previous entry of` M `*/`

$(f, i, j, (\alpha_i, \beta_j)) \leftarrow (f', i', j', (\alpha_{i'}, \beta_{j'}))$;

end

reverse(D'_A);

reverse(D'_B);

return $\{D'_A, D'_B\}$;

end

Algorithm 2: Traceback algorithm on a dynamic programming table M for retrieving the best local alignment between two sequences $D_A = a_{i_1}...a_{i_k}$ and $D'_B = b_{j_1}...b_{j_h}$ comprised in the back-translation graphs G_A and G_B respectively. The score of the best alignment is given by a cell $M^{f_b}[i_b, j_b, (\alpha^b_{i_b}, \beta^b_{j_b})]$ holding the highest value in M, which is the starting point of the traceback. The alignment is obtained in the form of two strings $D'_A = a'_1...a'_l$ and $D'_B = b'_1...b'_l$ of equal length l, resulting from D_A and D_B by inserting gaps. The pairs $(a'_h, b'_h), h = 1..l$ give the actual aligned symbols.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

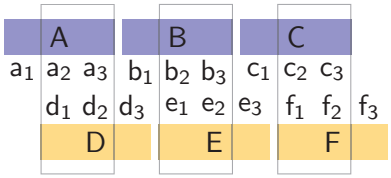

Figure 5.11: Generic example of alignment between frameshifted protein sequences. A, B, C, D, E, F are amino acids encoded respectively by the codons $a_1a_2a_3$, $b_1b_2b_3$ etc. In the depicted alignment, the DNA sequence encoding ABC is shifted by 1 relatively to the DNA sequence encoding DEF. Scoring functions that operate at the amino acid level, such as those proposed previously in [40] and [167], take into account only partially the alignment at the DNA level. As such, some of the codon overlaps (depicted here outside the regions emphasized by a rectangular marker) are ignored by the scoring function.

mutational patterns for coding sequences, based on a codon substitution model, with the aim of filtering out alignments between sequences that are unlikely to have common origins.

Mutation rates have been shown to vary within genomes, under the influence of different factors, including neighbor bases [23]. Consequently, a model where all base mismatches are equally penalized is oversimplified, and ignores possibly precious information about the context of the substitution. With the goal of being an instrument for retracing the sequence's evolution and revealing which base substitutions are more likely to occur within a given codon, this scoring system is defined on pairs of triplets (α, p, a), where α is a nucleotide symbol, p is its position in the codon, and a is the amino acid encoded by that codon, thus differentiating various contexts of a substitution. For non-ambiguous nucleotide symbols, there are 99 valid triplets out of the total of 240 hypothetical combinations, while 69 valid triplets exist when ambiguous symbols are used.

Unlike the scores previously proposed in [40] and [167] for aligning protein sequences directly with a frameshift, the scoring system discussed here takes into account all possible overlaps between codons that are frameshifted in the two aligned sequences. Taking as example the situation illustrated in Figure 5.11, the scores of [40] and [167] implicitly rely, at the nucleotide sequence level, on the pairs (a_2, d_1), (a_3, d_2), (b_2, e_1), (b_3, e_2), (c_2, f_1), (c_3, f_2), but ignore smaller codon overlaps, in this case (b_1, d_3) and (c_1, e_3). This may result in over-estimating the alignment score for some input sequences, as explained earlier in Section 4.2.1. This issue is addressed here by explicitly defining substitution scores for all nucleotides, with respect to an established reading frame, based on their position in the codon and the amino acid encoded by that codon.

5.3.1 Method for obtaining translation-dependent scores

In this section, the scores for non-ambiguous symbols are first defined, and constitute the basis for defining the substitution scores for ambiguous symbols, which are in fact used in practice. The pairwise alignment scores are computed for all possible pairs of valid triplets $(t_i, t_j) = ((\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j))$ as a classic log-odds ratio:

$$\sigma(t_i, t_j) = \lambda \log \frac{f_{t_i t_j}}{b_{t_i t_j}} \tag{5.12}$$

where $f_{t_i t_j}$ is the frequency of the $t_i \leftrightarrow t_j$ substitution in related sequences, and $b_{t_i t_j} = p(t_i)p(t_j)$ is the background probability. This scoring function is used in the algorithm as shown by equation (5.5)(b), where it appears as $\sigma(\alpha_A, \alpha_B)$, without explicitly mentioning the context – amino acid and position in the corresponding codon – of the paired nucleotides. These details

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

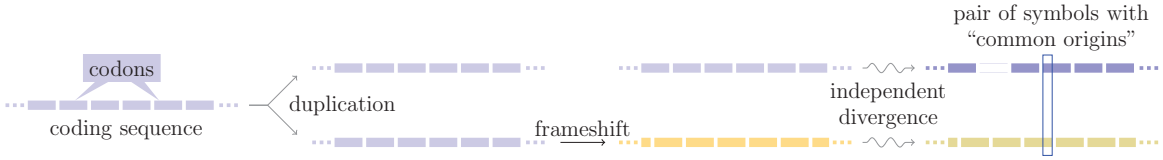

Figure 5.12: Sequence divergence by duplication and frameshift: a coding DNA sequence is duplicated, and one of the copies undergoes a frameshift mutation; subsequently, the two copies continue to diverge independently, via synonymous and non-synonymous point mutations, or full codon insertions and removals.

were omitted in equation (5.5) for generality (so that other scoring functions, that do not depend on the translation, can be used by the algorithm as well) and for notation simplicity.

The foreground probabilities $f_{t_i t_j}$ are defined according to the divergence scenario presented in Section 4.1 and depicted in Figure 5.12: a coding DNA sequence is duplicated, and one of the copies undergoes a frameshift mutation; subsequently, the two copies continue to diverge independently, via synonymous and non-synonymous point mutations, or full codon insertions and removals.

The insignificant amount of available real data that fits this hypothesis does not allow classical, statistical computation of the foreground and background probabilities. Therefore, instead of doing statistics on real data directly, the proposed scores rely on codon frequency tables and codon substitution models, either mechanistic [107] or empirically constructed [182]. These models were discussed earlier in Section 3.3.3, and basically express all the probabilities $P_{c_i,c_j}(\theta)$ that codon c_i is substituted by codon c_j after an evolutionary time θ, measured in expected number of nucleotide substitutions per codon.

Foreground probabilities

Once the codon substitution probabilities are obtained, $f_{t_i t_j}$ can be deduced in several steps. Basically, we first need to identify all pairs of codons with a common subsequence, that have a perfect semi-global alignment (for instance, codons CAT and ATG satisfy this condition, having the common subsequence AT; this example is further explained below). We then assume that the codons from each pair undergo independent evolution, according to the codon substitution model. For the resulting codons, we compute, based on all possible original codon pairs, $p((\alpha_i, p_i, c_i), (\alpha_j, p_j, c_j))$ – the probability that nucleotide α_i, located at position p_i of codon c_i, and nucleotide α_j, situated on position p_j of codon c_j have a common origin (equation (5.15)). From these, we can proceed to compute the probabilities $p((\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j))$, corresponding to the foreground probabilities $f_{t_i t_j}$, where $t_i = (\alpha_i, p_i, a_i)$ and $t_j = (\alpha_j, p_j, a_j)$. The computation formula combines the values $p((\alpha_i, p_i, c_i), (\alpha_j, p_j, c_j))$ corresponding to all codons c_i and c_j that encode the amino acids a_i and a_j respectively, as shown in equation (5.16).

In the following, $\mathbf{p}(\mathbf{c_i} \xrightarrow{\theta} \mathbf{c_j})$ stands for the probability of the event *codon c_i mutates into codon c_j in evolutionary time θ*, and is given by a codon substitution probability matrix $P_{c_i,c_j}(\theta)$, obtained as explained earlier in Section 3.3.3.

The notation $\mathbf{c_i}[\mathbf{I_i}] \equiv \mathbf{c_j}[\mathbf{I_j}]$ states that codon c_i restricted to the positions given by the interval I_i is a sequence identical to c_j restricted to the positions given by I_j. This is equivalent to having a word w obtained by "merging" the two codons. For instance, if $c_i = CAT$ and $c_j = ATG$, with their common substring being placed in $I_i = [2..3]$ and $I_j = [1..2]$ respectively, w is $CATG$. Table 5.3 illustrates the overlapping intervals that need to be taken into consideration in each of

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Reading frame	**Aligned codon positions**	I_A	I_B
+0	A: 123123	[1..3]	[1..3]
	B: 123123		
+1	A: 123123	[2..3]	[1..2]
	B: 123123	[1]	[3]
+2	A: 123123	[3]	[1]
	B: 123123	[1..2]	[2..3]
-1	A: 123123	[2..3]	[3..2]
	B: 321321	[1]	[1]
-2	A: 123123	[1..2]	[2..1]
	B: 321321	[3]	[3]
-3	A: 123123	[1..3]	[3..1]
	B: 321321		

Table 5.3: Overlapping codon intervals in each of the possible reading frame differences. Each case is labeled by an integer value representing the distance between the first position of the codons in the two sequences. Negative values (-1,-2,-3) are assigned to the cases when the second sequence is reversed and complemented.

the possible reading frame differences. Each case is labeled by an integer value representing the distance between the first position of the codons in the two sequences. Negative values (-1,-2,-3) are assigned to the cases when the second sequence is reversed and complemented.

We denote by $\mathbf{p}(\mathbf{c_i}[\mathbf{I_i}] \equiv \mathbf{c_j}[\mathbf{I_j}])$ the probability to have c_i and c_j, in the relation described above, and we compute it as the probability of the word w obtained by "merging" the two codons. This function should be symmetric, it should depend on the codon distribution, and the probabilities of all the words w of a given length should sum to 1. However, since we consider the case where the same DNA sequence is translated on two different reading frames, one of the two translated sequences would have an atypical composition. Consequently, the probability of a word w is computed as if the sequence had the known codon composition when translated on the reading frame imposed by the first codon, **or** on the one imposed by the second. This hypothesis can be formalized as:

$$p(w) = p(w \text{ on } rf_1 \text{ OR } w \text{ on } rf_2) = p^{rf_1}(w) + p^{rf_2}(w) - p^{rf_1}(w) \cdot p^{rf_2}(w) \tag{5.13}$$

where $p^{rf_1}(w)$ and $p^{rf_2}(w)$ are the probabilities of the word w in the reading frame imposed by the position of the first and second codon, respectively. This is computed as the products of the probabilities of the codons and codon pieces that compose the word w in the established reading frame. In the previous example, the probabilities of $w = CATG$ in the first and second reading frame are:

$$p^{rf_1}(CATG) = p(CAT) \cdot p(G**) = p(CAT) \cdot \sum_{c:c \text{ starts with } G} p(c)$$

$$p^{rf_2}(CATG) = p(**C) \cdot p(ATG) = \sum_{c:c \text{ ends with } C} p(c) \cdot p(ATG).$$

We denote the probability that codon c_i and c_j align with a frameshift f, overlapping in

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

their alignment on the intervals I_i and I_j, by $p_f^{I_i,I_j}(c_i, c_j)$, obtained as

$$p_f^{I_i,I_j}(c_i, c_j) = \sum_{c'_i, c'_j : c'_i[I_i] \equiv c'_j[I_j]} p(c'_i[I_i] \equiv c'_j[I_j]) \cdot p(c'_i \xrightarrow{\theta} c_i) \cdot p(c'_j \xrightarrow{\theta} c_j). \tag{5.14}$$

The notation $p_f^{I_i,I_j}$ is intentionally redundant, i.e. f can be deduced from I_i, I_j and viceversa.

Finally, the values of $p((\alpha_i, p_i, c_i), (\alpha_j, p_j, c_j))$ are computed as:

$$p((\alpha_i, p_i, c_i), (\alpha_j, p_j, c_j)) = p_f^{I_i,I_j}(c_i, c_j) \tag{5.15}$$

such that $p_i \in I_i$ and $p_j = I_j$ are aligned codon positions when c_i and c_j are aligned with the frameshift f. From these, obtaining the **foreground probabilities** is straightforward:

$$f_{t_i t_j} = p((\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j)) = \sum_{\substack{c_i \text{encodes } a_i, \\ c_j \text{encodes } a_j}} p((\alpha_i, p_i, c_i), (\alpha_j, p_j, c_j)). \tag{5.16}$$

Background probabilities

The **background probabilities** of (t_i, t_j), $b_{t_i t_j}$ can be simply expressed as the probability of the two symbols appearing independently in the sequences:

$$b_{t_i t_j} = b_{(\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j)} = \sum_{\substack{c_i \text{encodes } a_i, \\ c_j \text{encodes } a_j}} \pi_{c_i} \pi_{c_j}. \tag{5.17}$$

Substitution matrix for ambiguous symbols

The values $f_{t_i t_j}$ and $b_{t_i t_j}$ are substituted in (5.12) to obtain the pairwise alignment scores for all $(t_i, t_j) = ((\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j))$, where α_i, α_j are *non-ambiguous* nucleotide symbols, i.e. $\alpha_i, \alpha_j \in \{A, C, T, G\}$, p_i, p_j are their positions in the codon, and a_i, a_j are the amino acids encoded by those respective codons. However, *ambiguous* nucleotide symbols are used in practice to improve time and memory consumption while providing the same final results. The scores for ambiguous nucleotide symbol pairs are easily obtained as follows:

$$\sigma((\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j)) = \max_{x_i \in \alpha_i, x_j \in \alpha_j} \sigma((x_i, p_i, a_i), (x_j, p_j, a_j)) \tag{5.18}$$

where α_i is an ambiguous nucleotide symbol representing the possible nucleotides that can appear on position p_i of the codons that encode the amino acid a_i, and $x_i \in \alpha_i$ denotes the non-ambiguous nucleotide symbols represented by α_i. Basically, the score of pairing two ambiguous symbols is the maximum over all substitution scores for all pairs of nucleotides from the respective sets.

By using ambiguous symbols, less triplets are formed for each amino acid when compared with the non-ambiguous symbol case. 17 amino acids can be anti-translated as tri-mers with just one ambiguous symbol per position, while the others have two alternatives each of the three positions. Therefore, there are only 69 different triplets with ambiguity codes to be paired (as opposed to 99). It follows that the score matrix for non-ambiguous symbols has $99 \cdot 99 = 9801$ entries, while the number of entries in the matrix for ambiguous is $69 \cdot 69 = 4761$, thus requiring approximately twice less storage space necessary. In consequence, using ambiguous symbols for

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

encoding the protein's back-translation graph brings benefits not only to the size of the graphs and the complexity of the alignment algorithm, but also to the size of the matrix holding the pre-computed scores.

For the reconstruction of the non-ambiguous putative DNA sequences at traceback, the actual pair of nucleotides that have the highest substitution score from the sets corresponding to two paired ambiguous symbols is required. These are easily obtained for each pair of ambiguous symbols as

$$\tau((\alpha_i, p_i, a_i), (\alpha_j, p_j, a_j)) = arg \max_{x_i \in \alpha_i, x_j \in \alpha_j} \sigma((x_i, p_i, a_i), (x_j, p_j, a_j)). \tag{5.19}$$

Parametrization

This section presented a general framework that helps to compute a translation dependent scoring function for DNA sequence pairs, parametrized by a codon substitution model and an evolutionary time measured in expected number of mutations per codon. We consider that the sequences evolve independently, and the distance is relative to the original sequence.

5.3.2 Score evaluation

The score significance is estimated according to the Gumbel distribution, where the parameters λ and K are computed with the method proposed in [10,162] and presented earlier in Section 3.3.4.

We use two different score evaluation parameter sets for the forward alignment (where the two back-translated graphs that are aligned have the same translation sense) and the reverse complementary alignment (where one of the graphs is aligned with the reverse complementary of the other), because these are two independent cases with different score distributions.

In order to obtain a more refined evaluation of the alignments, we introduce (λ, K) parameters for estimating the score significance of alignment fragments inside which the reading frame difference is preserved. Therefore, there are eight (λ, K) parameters that help to evaluate the alignments (four for the forward alignment sense and four for the reverse complementary alignment sense):

- (λ_{FW}, K_{FW}) for the forward sense and (λ_{RC}, K_{RC}) for the reverse complementary sense respectively, that are used for evaluating the score of the whole alignment.
- (λ_{+i}, K_{+i}) for the forward sense and (λ_{-i}, K_{-i}) for the reverse complementary sense respectively, with $i \in \{0, 1, 2\}$ that are used for evaluating the scores of each alignment fragment within which the reading frame difference is preserved. This second evaluation aims at providing a measure of the actual contribution of each such fragment to the score of the alignment.

The parameters $(\lambda_{\pm i}, K_{\pm i})$ are estimated on alignments restricted to the respective reading frame difference, where further frameshifts are not allowed, while (λ_{FW}, K_{FW}) and (λ_{RC}, K_{RC}) are computed in a more flexible setup, where a limited number of frameshifts is accepted.

5.3.3 Comparison with classic amino acid and DNA substitution scores

In this section we discuss the behavior of the proposed scoring system when aligning protein sequences without a frameshift. Given their construction method, we expect the scores to reflect the amino acid similarities, but also to be influenced by similarities at the DNA level.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Translation dependent scoring matrix	**DNA**	**BLOSUM 62**
TDS$_M$ (0.1)	**0.86**	0.52
TDS$_M$ (0.3)	0.81	0.50
TDS$_M$ (0.5)	0.77	0.50
TDS$_M$ (0.7)	0.75	0.50
TDS$_M$ (1.0)	0.71	0.47
TDS$_E$	0.59	**0.88**

Table 5.4: Correlation coefficients of the translation dependent scores used on non-frameshifted amino acids, with BLOSUM scores and classic DNA scores. The correlation coefficients between several types of scores that can be used to align amino acids without a frameshift: i) expected amino acid pair scores obtained from codon alignment with a classic match/mismatch scoring scheme (denoted DNA); ii) expected amino acid pair scores obtained from the translation-dependent scoring matrices based on the mechanistic codon substitution model (denoted TDS_M); iii) expected amino acid pair scores obtained from the translation-dependent scoring matrices based on the empirical codon substitution model (denoted TDS_E); iv) BLOSUM matrices for amino acid sequence alignment.

To evaluate how our scores, used in non-frameshifted alignments, would relate to the classic scoring systems used by biological sequence comparison methods, we first compute, for each scoring matrix T corresponding to an evolutionary distance θ, the expected score for each amino acid pair, as:

$$T_{AA}(a_i, a_j) = \sum_{pos=1}^{3} \sum_{\substack{t,t':\exists \alpha_k,\alpha_l: \\ t=(\alpha_k,pos,a_i),t'=(\alpha_l,pos,a_j)}} p(t,t')T(t,t') \tag{5.20}$$

where

$$p(t,t') = \sum_{\substack{c:c \text{ encodes } a_i, \\ c[pos]=\alpha_k}} p(c) \cdot \sum_{\substack{c':c' \text{ encodes } a_j, \\ c'[pos]=\alpha_l}} p(c'). \tag{5.21}$$

Then, considering each amino acid pair as an observation, we compute the correlation coefficient of these expected scores and the BLOSUM matrices as given by [86].

We also evaluate the correlation with the expected amino acid pair scores obtained when the sequences are aligned using a classic nucleotide match/mismatch system. The latter expected amino acid pair scores are also obtained as weighted sums of scores, in a manner similar to the one described by equations (5.20) and (5.21), where the score for aligning two symbols has one of the three established values for match, transition mutation or transversion mutation. For these classic scores, we used the values $+5, -3, -4$ in the examples reported below, although we have not noticed any drastic changes when different sets of values are used.

The obtained correlation coefficients are reported in Table 5.3.3. These results suggest that the obtained translation dependent score matrices, either obtained from mechanistic or empirical codon substitution models, are a compromise between the "fully selective" BLOSUM matrices and the non-selective DNA scores.

On the one hand, the scores obtained using the mechanistic model do not make use of the selective pressure, and for this reason are more likely to be correlated with the classic DNA scores. On the other hand, the scores based on empirical codon substitution models reflect the constraints imposed by the similarity of the amino acids encoded by the codons. Hence, they show a strong correlation with the BLOSUM matrices when used without a frameshift.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 6

Seeds for heuristic similarity search in proteins with a frameshift

The quadratic time complexity of the alignment algorithm presented in Chapter 5 makes it unsuitable to large scale experiments. This chapter proposes a heuristic seed-based approach for speeding up the similarity search in large protein databases. Since the interest is in proteins that may share similarities at the coding DNA level with a frameshift, the main challenge in this context is to determine, directly on protein sequences, potential conserved regions of DNA that encode the proteins on different reading frames.

This chapter presents **preliminary work** concerning *a seeding concept adapted for the heuristic detection of potential frameshifted alignments of protein sequences*. The basis of the proposed seeding concept is introduced in Section 6.1.1 and formally explained in Section 6.1.2. The actual protocol of searching for frameshifted similarities between a query sequence and a database is presented in Section 6.2. Finally, Section 6.3 sets the foundation of a framework for analysing the efficiency of such seeds.

6.1 The concept of frameshift seeds

6.1.1 Informal introduction

The goal of the work presented in this chapter is to detect potential *frameshifted* alignments between protein sequences. It does not consider the non-frameshifted case, since the latter is a classic problem already addressed by several existing approaches.

One possible solution for detecting conserved regions is inspired by BLASTP, and consists of searching for two short protein sequences that have a high-scoring frameshifted alignment with respect to the scoring system discussed in Section 5.3. Nevertheless, the implementation of this approach is generally quite costly, since it relies on a more elaborate mechanism than simple hashing of short subsequences and direct retrieval of similar regions based on their hash key. Also, searching for similarities on 5 different reading frames would require 5 different indexes. We investigate here the possibility of using simpler seeding concepts, such as *spaced seeds* [133] or *subset seeds* [110] (presented earlier in Chapter 3, Section 3.4.3 and Section 3.4.5), that allow *direct hashing* and *using the same index for several reading frames* while also capturing, albeit less accurately, relevant similarities between frameshifted protein sequences. To do so, we first examine the conservation patterns at the DNA level and subsequently devise the means to express it at the protein level.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Reading frame	Aligned codon positions
+1	A: 123123 B: 123123
+2	A: 123123 B: 123123
−1	A: 123123 B: 321321
−2	A: 123123 B: 321321
−3	A: 123123 B: 321321

Table 6.1: Paired codon positions in frameshifted DNA alignments. Each frameshift scenario is identified by an integer value, and negative identifiers $(-1, -2, -3)$ are assigned to the cases when the second sequence is reversed and complemented. The 3rd positions of each codon, represented in gray, generally varies significantly within the set of codons for each amino acid, and for this reason cannot be considered as conservation evidence in an alignment. Positions 1 and 2 of the codons are more stable, and, when matched in two aligned sequences (cases represented in blue), may be used as anchors for inferring similar regions.

As can be deduced from Figure 1.12 (Chapter 1), the codons encoding a given amino acid tend to be identical on positions 1 and 2, and vary on the 3rd position. Since the back-translation graph of a protein implicitly incorporates all possible codons for every amino acid in the sequence, it follows that every 3rd position of this graph is likely to be ambiguous, i.e. present more than one possible nucleotide. As explained in the previous chapter, this is true for 18 out of the 20 amino acids. Moreover, 8 of the 20 amino acids are encoded by at least 4 codons and therefore their corresponding subgraphs have all possible nucleotides on their 3rd position. In conclusion, the 3rd position of any back-translation graph is generally unreliable when searching for DNA sequence similarities, and any evidence of sequence relatedness should be obtained by examining the nucleotides appearing on the 1st and 2nd positions of the codons.

This pattern, which ignores the 3rd codon position due to its instability, has already been successfully used for similarity search in coding DNA [26, 216]. Nevertheless, in the case of frameshifted alignments, the 3rd codon position of one sequence is aligned to the 1st or 2nd codon position of the other sequence. Table 6.1 enumerates the 5 possible frameshift situations: two for forward alignment and three corresponding to alignments when the second sequence is reversed and complemented, i.e. the first sequence is aligned to the *opposite strand* of the second. In each case, the pairs of codon positions that can provide evidence of sequence similarity at the DNA level is represented in bold. With the exception of the -2 scenario, only one position out of 3 for each codon can provide sufficient information (it is not ambiguous nor aligned to an ambiguous position of the other sequence) for inferring conservation. In terms of spaced seeds, this corresponds to the similarity pattern $1{*}{*}1{*}{*}1{*}{*}1...$ applied on the underlying DNA sequences.

However, it is preferred to recognize conserved regions directly on the protein sequences, in order to prevent from performing back-translation operations on proteins that will not be involved in any significant alignment. The transition from the DNA similarity pattern that searches for one in three matching position to the protein similarity pattern is straightforward: the amino acids can be "tagged" with the set of nucleotide(s) appearing on the first or second

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Codon position	Nucleotide	Code	Amino acids
1	A	0	I,M,T,N,K,S,R
	C	1	L,P,H,Q,R
	G	2	V,A,D,E,G
	T	3	F,L,S,Y,C,W
2	A	0	Y,H,Q,N,K,D,E
	C	1	S,P,T,A
	G	2	C,W,R,S,G
	T	3	F,L,I,M,V
−1	A	0	F,L,S,Y,C,W
(1 in reverse complementary)	C	1	V,A,D,E,G
	G	2	L,P,H,Q,R
	T	3	I,M,T,N,K,S,R
−2	A	0	F,L,I,M,V
(2 in reverse complementary)	C	1	C,W,R,S,G
	G	2	S,P,T,A
	T	3	Y,H,Q,N,K,D,E

Table 6.2: Sets of amino acids with the same nucleotide on a certain position of their codons (either first or second position). The 3rd codon position is ignored on purpose, since it does not provide reliable information, as explained in Section 6.1.1. This labeling does not *partition* the set of amino acids, i.e. some amino acids (L, R, S) have multiple labels. However, less than 12% of all amino acids in a protein sequence are expected to be multiply labeled.

position of their codons, as shown in Table 6.2, and the hashing function takes into account, for a given sequence fragment, the representative nucleotide of each amino acid in the fragment. Similarities are detected via identical hash values that take into account different codon positions in the two sequences, as suggested by Table 6.1.

Since amino acids are organized in sets, and members of a set are considered to behave identically, this seeding technique seems related to the concept of *subset seeds* [110]. However, there are several fundamental differences between the approach introduced here and the original subset seeds. Basically, the pattern of subset seeds is defined over an alphabet $\mathcal{B}$, where each symbol in $\mathcal{B}$ accepts a certain set of mismatches between aligned amino acids, with the additional constraints that every symbol in $\mathcal{B}$ accepts matches, and there is a symbol in $\mathcal{B}$ accepting *only* matches. These two conditions are not true in the setup presented above: first, an amino acid does not necessarily align with itself with a frameshift, and second, because of the asymmetry of the frameshifted alignments, two amino acids that match with a frameshift may not match with the same frameshift if swapped. Several examples illustrating these situations are given in Figure 6.1. Nevertheless, as will be shown in Section 6.3, the theory behind subset seeds *can* be used in this context, provided that the representation of the target alignments is adapted to comply with the subset seed restrictions.

6.1.2 Definition

Let $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ be the alphabet of amino acid symbols. According to Table 6.2, we can define a function $code_p : \Sigma \to \{0, 1, 2, 3\}^{\{1,2\}}$, which associates to each amino acid one or two numeric values. These values express the nucleotide

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

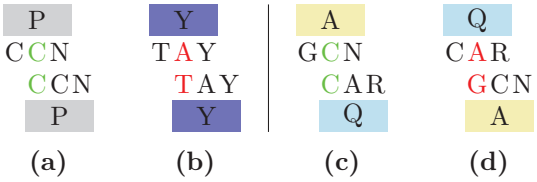

Figure 6.1: Illustrations of the asymmetry of frameshifted alignment at the amino acid level. For example, in a +1 frameshifted alignment, only 6 of the 20 amino acids (P – illustrated in **(a)**, F, L, N, K, G) match with themselves at the DNA level in a +1 frameshift, while most of them do not, as is the case of the amino acid Y shown in **(b)**. Additionally, if two different amino acids match at the DNA level in a +1 frameshifted alignment, the converse is not necessarily true, as illustrated by the pair of amino acids A and Q in **(c)** and **(d)**.

Figure 6.2: Frameshifted alignment between two short amino acid sequences: NVA and MWP. The alignment at the DNA level highlights nucleotide matches occurring every 3rd position ($1**1**1$).

or nucleotides appearing on the position indicated by $p \in \{1, 2, -1, -2\}$ in at least one of the codons that encode the amino acid. Negative values of p indicate that the codons are reversed and complemented. For example, $code_2(M) = \{3\}$ (standing for T, which appears on the second position of codon ATG that encodes M) and $code_{-1}(L) = \{0, 2\}$ (standing for A – the complementary of T, and G – the complementary of C, both appearing on the first position of codons CTN and TTR which encode L). The codes for each amino acid are given by Table 6.3.

Small local frameshifted alignments similarities between two sequences $S = s_0...s_{m-1}$ and $T = t_0...t_{n-1}$ can be detected by identifying words of length k in S and T, namely $S_{i..i+k-1} = s_i...s_{i+k-1}$ and $T_{j..j+k-1} = t_j...t_{j+k-1}$, such that

$$code_p(s_{i+d}) \cap code_q(t_{j+d}) \neq \emptyset, \forall d \in \{0, ..., k-1\}. \tag{6.1}$$

This condition basically states that, in the back-translation of $S_{i..i+k-1}$ and $T_{j..j+k-1}$, there are nucleotides on the pth and qth position of each codon which are respectively identical. For instance, let $k = 3$ and let the words $S_{i..i+k-1}$ and $T_{j..j+k-1}$ be the amino acid sequences NVA and MWP respectively. Then, according to Table 6.3, $code_2(N) = code_1(M) = \{0\}$, $code_2(V) = code_1(W) = \{3\}$, $code_2(A) = code_1(P) = \{1\}$, and this match corresponds to the frameshifted alignment in Figure 6.2. Note that the definition still holds when hashing gapped k-mers instead of contiguous words.

Any of the 5 frameshifted alignment scenarios displayed in Table 6.1 is detectable by a certain pair of codon positions p and q, that select the nucleotides to match in a certain reading frame difference, as shown in table Table 6.4.

In order to identify pairs of matching k-words, we define $\mathcal{H} = \{H_p : \Sigma^k \rightarrow \mathbf{N}^{\{1,...,2^k\}}\}$ as a family of hash functions,

$$H_p(w_0...w_{k-1}) = \bigcup \left\{ \sum_{i=0}^{k-1} c_i \cdot 4^i | c_i \in code_p(w_i) \right\} \tag{6.2}$$

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Amino acid	**Codons**	$\mathbf{code}_1$	$\mathbf{code}_2$	$\mathbf{code}_{-1}$	$\mathbf{code}_{-2}$
A	GCN	$\{2\}$	$\{1\}$	$\{1\}$	$\{2\}$
C	TGY	$\{3\}$	$\{2\}$	$\{0\}$	$\{1\}$
D	GAY	$\{2\}$	$\{0\}$	$\{1\}$	$\{3\}$
E	GAR	$\{2\}$	$\{0\}$	$\{1\}$	$\{3\}$
F	TTY	$\{3\}$	$\{3\}$	$\{0\}$	$\{0\}$
G	GGN	$\{2\}$	$\{2\}$	$\{1\}$	$\{1\}$
H	CAY	$\{1\}$	$\{0\}$	$\{2\}$	$\{3\}$
I	ATH	$\{0\}$	$\{3\}$	$\{3\}$	$\{0\}$
K	AAR	$\{0\}$	$\{0\}$	$\{3\}$	$\{3\}$
L	CTN, TTR	$\{1,3\}$	$\{3\}$	$\{0,2\}$	$\{0\}$
M	ATG	$\{0\}$	$\{3\}$	$\{3\}$	$\{0\}$
N	AAY	$\{0\}$	$\{0\}$	$\{3\}$	$\{3\}$
P	CCN	$\{1\}$	$\{1\}$	$\{2\}$	$\{2\}$
Q	CAR	$\{1\}$	$\{0\}$	$\{2\}$	$\{3\}$
R	AGR, CGN	$\{0,1\}$	$\{2\}$	$\{2,3\}$	$\{1\}$
S	AGY, TCN	$\{0,3\}$	$\{1,2\}$	$\{0,3\}$	$\{1,2\}$
T	ACN	$\{0\}$	$\{1\}$	$\{3\}$	$\{2\}$
V	GTN	$\{2\}$	$\{3\}$	$\{1\}$	$\{0\}$
Y	TAY	$\{3\}$	$\{0\}$	$\{0\}$	$\{3\}$
W	TGG	$\{3\}$	$\{2\}$	$\{0\}$	$\{1\}$

Table 6.3: The values of the functions $code_p : \Sigma \to \{0,1,2,3\}^{\{1,2\}}$, where $p \in \{1,2,-1,-2\}$ and $\Sigma = \{A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y\}$ is the alphabet of amino acid symbols. $code_p(\cdot)$ associates to each amino acid in Σ one or two numeric values, expressing the nucleotide(s) that appear on position p of the codons encoding the amino acid. Negative values of p indicate that the codons are reversed and complemented. The nucleotides A, C, G, T are encoded as 0, 1, 2, 3 respectively.

Reading frame	p	q
$+1$	2	1
$+2$	1	2
-1	1	-1
-2	1	-2
	2	-1
-3	2	-2

Table 6.4: Codon positions that need to be matched for detecting similarities on each frameshift. The first column identifies each frameshift, illustrated earlier in Table 6.1, while p and q indicate the codon position to take into account in the first and second sequence respectively.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

with $p \in \{1, 2, -1, -2\}$ and $code_p$ defined as above. For a word $w \in \Sigma^k$, $w = w_0...w_{k-1}$, $H_p(w)$ basically expresses, in the same manner as spaced seeds, which nucleotide symbols appear at the codon positions indicated by p in putative DNA sequences of w. For example,

$$\begin{aligned} H_1(APY) &= \{2 \cdot 4^0 + 1 \cdot 4^1 + 3 \cdot 4^2\} = \{36\} \\ H_2(SAL) &= \{1 \cdot 4^0 + 2 \cdot 4^1 + 3 \cdot 4^2, 2 \cdot 4^0 + 2 \cdot 4^1 + 3 \cdot 4^2\} = \{41, 42\}. \end{aligned}$$

Although, in theory, a k-mer could have a large exponential number (2^k) of corresponding keys in the index, in practice the expected number of keys per k-mer is about $2^{0.216k}$ according to Table 6.3 and assuming the amino acid composition of Table 5.1. For example, the number of keys per 4-mer is about 1.82, and 8-mers are expected to have approximately 3.3 keys in the index. Alternatively, a single key can be for each k-mer, probabilistically according to codon frequencies, with the price of a sensitivity decrease.

To identify similarities between sequences $S = s_0...s_{m-1}$ and $T = t_0...t_{n-1}$ using the hashing functions in $\mathcal{H}$, all words of length k in S and T are hashed, and those with the same hash value are selected. As such, relation (6.1) defining when two k-words $S_{i..i+k-1} = s_i...s_{i+k-1}$ and $T_{j..j+k-1} = t_j...t_{j+k-1}$ match becomes

$$H_p(S_{i..i+k-1}) \cap H_q(T_{j..j+k-1}) \neq \emptyset \tag{6.3}$$

with p and q chosen as indicated by Table 6.4 for a particular frameshift. In the example above (Figure 6.2), where the alignment with a +1 frameshift requires that $p = 2$ and $q = 1$, $H_2(NVA) = \{0 \cdot 4^0 + 3 \cdot 4^1 + 1 \cdot 4^2\} = \{28\}$ and $H_1(MWP) = \{0 \cdot 4^0 + 3 \cdot 4^1 + 1 \cdot 4^2\} = \{28\}$ according to (6.2), hence $H_2(NVA) \cap H_1(MWP) \neq \emptyset$.

6.2 Similarity search procedure

As can be deduced from Table 6.4, one of the sequences can be processed using only H_1 and H_2, while for the other sequence all four hash functions must be used in order to determine possible frameshifted alignments in the forward and reverse complementary sense.

Let $\mathcal{D} = \{S_1, ..., S_N\}$ be a database of protein sequences and Q a query (protein sequence) to be searched in $\mathcal{D}$, with $S_1, ..., S_N, Q \in \Sigma^+$. Note that any processing described below relies on the representation as *protein sequences*, i.e. $S_1, ..., S_N, Q$ are strings over the alphabet $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ of amino acids symbols, not yet back-translated. As such, a k-mer appearing at position d of a protein sequence $S = s_0...s_{n-1} \in \mathcal{D} \cup \{Q\}$ is a word $w \in \Sigma^k$, with $w_j = s_{d+j}, \forall j = 0..k-1$.

Prior to the actual query search, $\mathcal{D}$ is **indexed** using the hashing functions H_1 and H_2. For an established k, all the k-mers appearing in the sequences S_i are hashed using these functions. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the indexes resulting from processing $\mathcal{D}$ with H_1 and H_2 respectively. Each index $\mathcal{I}_p$ ($p = 1, 2$) contains the set of *keys ranging from 0 to* 4^k, and every key h points to a *list of pairs* (i, d) giving the identifier i of the sequence $S_i \in \mathcal{D}$ and the actual position d within S_i where there is a k-mer w with $h \in H_p(w)$. This list is denoted $\mathcal{I}_p[h]$. The indexing step is performed only once for a database, and the index is stored in order to be used for subsequent queries.

To detect potential frameshifted alignments between a query sequence Q and the sequences S_i in $\mathcal{D}$, Q is processed, in *four passes*, with the 4 hashing functions H_1, H_2, H_{-1}, H_{-2}. All k-mers appearing in Q are hashed using these functions, and the resulted keys are searched in $\mathcal{I}_1$ and $\mathcal{I}_2$ accordingly with Table 6.4. Namely, let $w = w_0...w_{k-1}$ be a k-mer appearing at some position d' in Q. Then, hits are identified as follows:

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

- Let $h_1 = H_1(w)$.
  $\mathcal{I}_2[h_1] = \{(i, d)\}$ gives the list of potential local similarities between Q (at position d') and the sequences $S_i \in \mathcal{D}$ (at their respective positions d) on the $+1$ **frameshift**.

- Let $h_2 = H_2(w)$.
  $\mathcal{I}_1[h_2] = \{(i, d)\}$ gives the list of potential local similarities between Q (at position d') and the sequences $S_i \in \mathcal{D}$ (at their respective positions d) on the $+2$ **frameshift**.

- Let $h_{-1} = H_{-1}(w)$.
  $\mathcal{I}_1[h_{-1}] = \{(i, d)\}$ gives the list of potential local similarities between Q (at position d') and the sequences $S_i \in \mathcal{D}$ (at their respective positions d) on the -1 **frameshift**.
  $\mathcal{I}_2[h_{-1}] = \{(i, d)\}$ gives the list of potential local similarities between Q (at position d') and the sequences $S_i \in \mathcal{D}$ (at their respective positions d) on the -2 **frameshift**.

- Let $h_{-2} = H_{-2}(w)$.
  $\mathcal{I}_1[h_{-1}] = \{(i, d)\}$ gives the list of potential local similarities between Q (at position d') and the sequences $S_i \in \mathcal{D}$ (at their respective positions d) on the -2 **frameshift**.
  $\mathcal{I}_2[h_{-1}] = \{(i, d)\}$ gives the list of potential local similarities between Q (at position d') and the sequences $S_i \in \mathcal{D}$ (at their respective positions d) on the -3 **frameshift**.

Both single and multiple hit strategies (provided that multiple hits appear on the same frameshift) can be used for identifying significant local similarities. When significant hits are detected in a sequence $S_i \in \mathcal{D}$, the query Q and S_i are submitted to the alignment algorithm of Chapter 5 for validation.

6.3 Analysis

The sensitivity of such seeds, i.e. their capacity to detect significant frameshifted protein alignments, can be evaluated using the IEDERA software [110–112]. As explained earlier in Section 3.4.4, *Iedera* is designed for spaced/subset seeds, and relies on a probabilistic model of target alignments with respect to which it computes the theoretical sensitivity of a seed.

To comply with the representation of target alignments required by IEDERA, ungapped frameshifted alignments are defined by:

Definition 13. *An* ungapped protein alignment with a frameshift $f \in \{+1, +2, -1, -2, -3\}$ *(see Table 6.1) is defined as a word* $A = a_1...a_n$ *over the alphabet* $\mathcal{A} = \{0, 1\}$*, where 1 (match) denotes a pair of aligned amino acids* x *and* y *such that* $code_p(x) \cap code_q(y) \neq \emptyset$*, as explained earlier in relation* (6.1)*, and 0 (mismatch) denotes a pair of aligned amino acids* x *and* y *such that* $code_p(x) \cap code_q(y) = \emptyset$*.*

Additionally, we remind that a *spaced seed* is a word π over the alphabet $\mathcal{B} = \{*, 1\}$, such that 1 accepts a match and * accepts either a match or a mismatch. Note that, from this point forward, **matches and mismatches are *not* considered with respect to individual nucleotides**. As mentioned by Definition 13, two amino acids x and y "match" with a frameshift f if and only if $code_p(x) \cap code_q(y) \neq \emptyset$, i.e. if a nucleotide appearing at position p of a codon encoding x matches (in the classic sense) a nucleotide appearing on position q of a codon encoding y, with p and q chosen as indicated earlier in Table 6.4. With this definition for "matches" between amino acids, the similarities are in fact considered **at the codon level**. Consequently, it makes sense to **model frameshifted codon alignments** and evaluate seeds with respect to

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

such alignments. The seeds can then be used directly on amino acids sequences thanks to the hashing function given by (6.2), which implicitly takes into account matches at the codon level.

In order to compute the sensitivity of a seed, IEDERA requires as input, besides the seed pattern, a *foreground model* of significant frameshifted alignments, based on which the probability to hit a target alignment is determined, as well as a *background model of random alignments*. Both models are described below.

6.3.1 Modeling frameshifted alignments

With the exception of the -3 frameshift, where aligned amino acids overlap completely, allowing successive symbols of the alignment representation to be generated independently of each other according to their respective probabilities, all other frameshifted alignments $(+1, +2, -1, -2)$ have some degree of dependency of the symbol at position i on the symbol at position $i-1$, since 1 codon in one sequence is partly aligned with 2 codons in the other sequence.

In the following, let us explain the model for significant frameshifted alignments for the $+1$ frameshift, where the first codon position in one sequence is aligned to the second codon position in the other sequence: $\begin{smallmatrix}\texttt{123}\\ \texttt{123}\end{smallmatrix}$. The other frameshift cases can then be handled in a similar manner, and therefore a detailed description of their corresponding models is omitted.

Notations

Since we are interested in similarities occurring at the codon level, codons are explicitly represented from this point forward by their 3 nucleotide symbols, for example ABC. The following **probabilities, obtained as explained in Chapter 5, Section 5.3 will be involved in the definition of the model**:

- The occurrence probability in a coding sequence of each codon ABC, denoted $p(\texttt{ABC})$;
- The probability that codon c_i and c_j align with a frameshift f, overlapping in their alignment on the intervals I_i and I_j, given by relation (5.14) of Section 5.3, Chapter 5 and originally denoted $p_f^{I_i,I_j}(c_i, c_j)$; here we use a notation that expresses visually the frameshift and the aligned intervals, namely $p\left(\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}\right)$ for the $+1$ frameshift or $p\left(\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}\right)$ for the $+2$ frameshift; note also that, by definition, $p\left(\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}\right) = p\left(\begin{smallmatrix}\texttt{DEF}\\ \texttt{ABC}\end{smallmatrix}\right)$.

Naive foreground model for alignments with a +1 frameshift

A straightforward model of significant alignments with a $+1$ frameshift can be obtained as follows. A *state* represents a pair of codons aligned with the $+1$ frameshift, denoted $\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}$. *Transitions* from a state $\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}$ to another state $\begin{smallmatrix}\texttt{GHI}\\ \texttt{JKL}\end{smallmatrix}$ have the meaning that, in an alignment with a $+1$ frameshift, codon GHI follows ABC in one sequence and JKL follows DEF in the other, producing the alignment $\begin{smallmatrix}\texttt{ABCGHI}\\ \texttt{DEFJKL}\end{smallmatrix}$. As such, symbol F is aligned to symbol G, and the transition probability should depend on the chance to have these two symbols aligned in significant alignments. The *emissions* that occur when transitioning to a state $\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}$ only depend on whether $code_2(\texttt{ABC}) = code_1(\texttt{DEF})$, i.e. whether the nucleotide symbols B and D match.

More formally, this model is defined by:

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

- A set of states $Q = \{S\} \cup \left\{ \begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix} \right\}$, where S is the initial state and (ABC, DEF) enumerate all possible pairs of codons; the size of Q is $C^2 + 1$, where C is the number of possible codons encoding for amino acids; note that STOP codons are not considered here;
- A function $\iota : Q \rightarrow [0, 1]$ giving the probability of each state in Q to be the initial state, with

$$\iota[S] = 1 \tag{6.4}$$

$$\iota\left[\begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix} \right] = 0, \forall \begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix} \in Q \setminus \{S\}; \tag{6.5}$$

- A transition probability function $\delta : Q \times Q \rightarrow [0, 1]$:

$$\delta\left(S, \begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix}\right) = p\left(\begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix} \right) \tag{6.6}$$

$$\delta\left(\begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix}, \begin{smallmatrix} \texttt{GHI} \\ \texttt{JKL} \end{smallmatrix} \right) = \frac{p\left(\begin{smallmatrix} \texttt{GHI} \\ \texttt{DEF} \end{smallmatrix} \right)}{\sum_{x,y,z} p\left(\begin{smallmatrix} \texttt{xyz} \\ \texttt{DEF} \end{smallmatrix} \right)} \cdot \frac{p\left(\begin{smallmatrix} \texttt{GHI} \\ \texttt{JKL} \end{smallmatrix} \right)}{\sum_{x,y,z} p\left(\begin{smallmatrix} \texttt{GHI} \\ \texttt{xyz} \end{smallmatrix} \right)}. \tag{6.7}$$

  where x, y, z are nucleotide symbols, such that xyz instantiate all possible non-STOP codons; relation (6.7) basically states that the transition probability from $\begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix}$ to $\begin{smallmatrix} \texttt{GHI} \\ \texttt{JKL} \end{smallmatrix}$ depends on the probability that, in significant +1 frameshifted alignments, the last nucleotide of codon DEF be aligned to the first nucleotide of codon GHI among all possible alignment choices for DEF, **and** the last two nucleotides of codon GHI be aligned to the first two nucleotide of codon JKL among all possible alignment choices for GHI;
- An emission alphabet $\mathcal{A} = \{0, 1\}$, where 0 denotes a "mismatch" and 1 denotes a "match", in the sense given by Definition 13;
- An emission probability function $\beta : Q \times Q \times \mathcal{A} \rightarrow [0, 1]$, such that:

$$\beta\left(q, \begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix}, 1\right) = \begin{cases} 1, \text{ if } \texttt{B} = \texttt{D}; \\ 0, \text{ otherwise} \end{cases}, \forall q \in Q \tag{6.8}$$

$$\beta\left(q, \begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix}, 0\right) = \begin{cases} 0, \text{ if } \texttt{B} = \texttt{D}; \\ 1, \text{ otherwise} \end{cases}, \forall q \in Q. \tag{6.9}$$

The model is depicted in Figure 6.3. It has 3722 states ($61^2 + 1$) if unambiguous codons are considered. This number can be reduced to 530 by the use of ambiguous codons (see Table 5.1.2 of Chapter 5) in its definition. Nevertheless, the number of states remains very large and is likely to have considerable effects on the speed of the algorithms computing the seed sensitivity.

Improved foreground model for alignments with a +1 frameshift

In the following, we define a probabilistic model of significant alignments with a +1 frameshift that is equivalent to the one described above, but has a significantly smaller number of states and transitions. This is done by **grouping states according to symbols appearing on specific codon positions**, and adjusting transition and emission probabilities accordingly, as explained below.

Let $\begin{smallmatrix} \texttt{ABC} \\ \texttt{DEF} \end{smallmatrix}$ and $\begin{smallmatrix} \texttt{GHI} \\ \texttt{JKL} \end{smallmatrix}$ be two states of the probabilistic model described above. The following pairs of symbols influence the transition and emission probabilities:

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

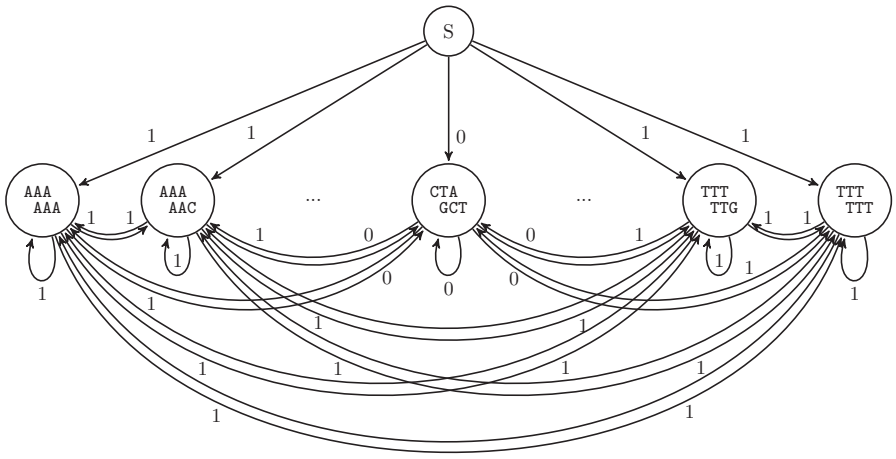

Figure 6.3: Model for generating gapped alignments on a +1 frameshift: 3722 states.

- `F` and `G` are involved in the probability of transition from $\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}$ to $\begin{smallmatrix}\texttt{GHI}\\ \texttt{JKL}\end{smallmatrix}$ according to relation (6.7);
- `B` and `D` dictate the emission probabilities when entering $\begin{smallmatrix}\texttt{ABC}\\ \texttt{DEF}\end{smallmatrix}$, according to (6.8) and (6.9); in the same manner, `H` and `J` dictate the emission probabilities when entering $\begin{smallmatrix}\texttt{GHI}\\ \texttt{JKL}\end{smallmatrix}$.

Based on these observations, the number of states can be reduced by grouping them according to these symbol positions in their representative codon alignments. As such, states have the form $\begin{smallmatrix}\texttt{Ann}\\ \texttt{DnF}\end{smallmatrix}$, representing all pairs of codons aligned with a +1 frameshift, where the first codon starts with the symbol `A` and the second codon has the symbol `D` on its first position and the symbol `F` on its last position. The codon positions where the nucleotide symbol is not fixed are marked by `n`. Note that `B` (the second position of the first codon in the aligned pair) is not fixed, and matches/mismatches are emitted with probabilities depending only on `D` and codon frequencies, as explained below.

For example, let Gnn be all the *ambiguous* codons starting with the nucleotide G, AnY represent all the *ambiguous* codons having the nucleotide A on the first position and the ambiguous symbol Y (standing for pyrimidines C and T) on the last. Then the state labeled $\begin{smallmatrix}\texttt{Gnn}\\ \texttt{AnY}\end{smallmatrix}$ denotes the following set of pairs of codons aligned with a +1 frameshift: $\begin{smallmatrix}\texttt{GTN}\\ \texttt{AAY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GCN}\\ \texttt{AAY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GAR}\\ \texttt{AAY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GAY}\\ \texttt{AAY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GGN}\\ \texttt{AAY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GTN}\\ \texttt{AGY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GCN}\\ \texttt{AGY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GAR}\\ \texttt{AGY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GAY}\\ \texttt{AGY}\end{smallmatrix}$, $\begin{smallmatrix}\texttt{GGN}\\ \texttt{AGY}\end{smallmatrix}$.

According to Table 5.1.2 of Chapter 5, there are 4 symbol posibilities for the first position of the codon (meaning that codons can be grouped according to their first symbol in Ann, Cnn, Gnn, Tnn) and 15 valid `DnF` variants among the possible ambiguous codons (namely AnG, AnH, AnY, AnR, AnN, CnY, CnR, CnN, GnR, GnY, GnN, TnG, TnR, TnY, TnN). Consequently, considering the states corresponding to the possible $\begin{smallmatrix}\texttt{Ann}\\ \texttt{DnF}\end{smallmatrix}$ combinations and adding a start state S as before, the number of states for this automaton becomes $4 \cdot 15 + 1 = 61$.

The improved model is depricted in Figure 6.4 and is formally defined by:

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

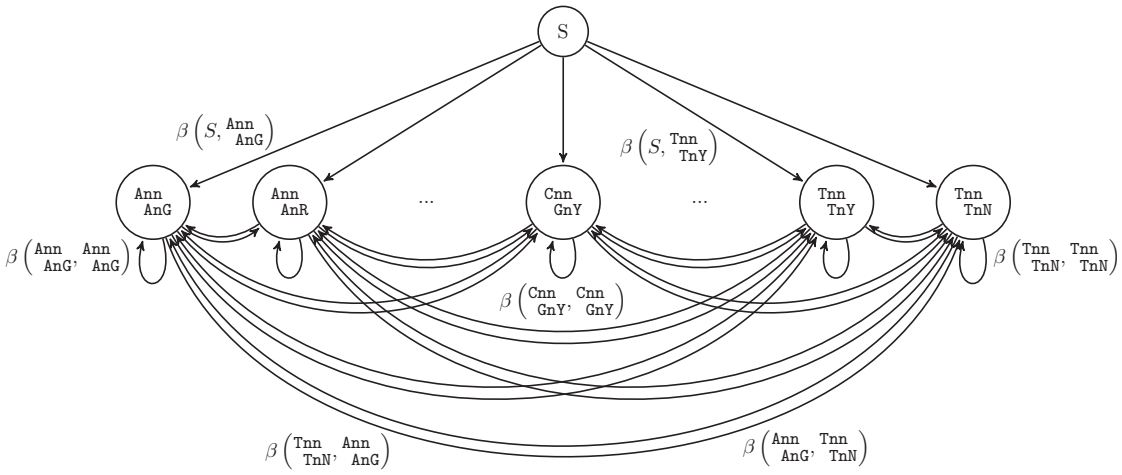

Figure 6.4: Improved model for generating gapped alignments on a +1 frameshift: 61 states.

- A set of states $Q = \{S\} \cup \left\{ \begin{smallmatrix} \texttt{Ann} \\ \texttt{DnF} \end{smallmatrix} | \texttt{A,D} \in \{A,C,G,T\}, \texttt{F} \in \{G,R,Y,H,N\} \right\}$, where S is the initial state, and $\begin{smallmatrix} \texttt{Ann} \\ \texttt{DnF} \end{smallmatrix}$ represents all alignments with a +1 frameshift of codons `Ann` starting with the symbol `A`, and codons `DnF` starting with `D` and ending with `F`;

- A function $\iota : Q \to [0,1]$ giving the probability of each state in Q to be the initial state, with

$$\iota[S] = 1 \tag{6.10}$$

$$\iota\left[\begin{smallmatrix} \texttt{Ann} \\ \texttt{DnF} \end{smallmatrix}\right] = 0, \forall \begin{smallmatrix} \texttt{Ann} \\ \texttt{DnF} \end{smallmatrix} \in Q \setminus \{S\}; \tag{6.11}$$

- A transition probability function $\delta : Q \times Q \to [0,1]$:

$$\delta\left(S, \begin{smallmatrix} \texttt{Ann} \\ \texttt{DnF} \end{smallmatrix}\right) = \sum_{b,c,e} p\left(\begin{smallmatrix} \texttt{Abc} \\ \texttt{DeF} \end{smallmatrix}\right) \tag{6.12}$$

$$\delta\left(\begin{smallmatrix} \texttt{Ann} \\ \texttt{DnF} \end{smallmatrix}, \begin{smallmatrix} \texttt{Gnn} \\ \texttt{JnL} \end{smallmatrix}\right) = \sum_{e,h,i,k} \left(\frac{p\left(\begin{smallmatrix} \texttt{Ghi} \\ \texttt{DeF} \end{smallmatrix}\right)}{\sum_{x,y,z} p\left(\begin{smallmatrix} \texttt{xyz} \\ \texttt{DeF} \end{smallmatrix}\right)} \cdot \frac{p\left(\begin{smallmatrix} \texttt{Ghi} \\ \texttt{JkL} \end{smallmatrix}\right)}{\sum_{x,y,z} p\left(\begin{smallmatrix} \texttt{Ghi} \\ \texttt{xyz} \end{smallmatrix}\right)} \right). \tag{6.13}$$

  where e, h, i, x, y, z are ambiguous nucleotide symbols, $e, h, x, y \in \{A,C,G,T\}$ and $i, z \in \{G,R,Y,H,N\}$, such that `DeF` and `Ghi` instantiate possible ambiguous codons with the fixed symbols `D`, `F` and `G` respectively, and `xyz` ennumerates all possible ambiguous codons encoding for amino acids; relation (6.13) is obtained from relation (6.7) by taking into account the hypothesis of (6.7) for all combinations of codons corresponding to groups `Ann`, `DnF`, `Gnn` and `JnL`;

- An emission alphabet $\mathcal{A} = \{0,1\}$, defined as above;

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

- An emission probability function $\beta : Q \times Q \times \mathcal{A} \to [0, 1]$, such that:

$$\beta\left(q, \begin{smallmatrix}\texttt{Ann}\\ \texttt{DnF}\end{smallmatrix}, 1\right) = \frac{\sum_{c,e} p\left(\begin{smallmatrix}\texttt{ADc}\\ \texttt{DeF}\end{smallmatrix}\right)}{\sum_{b,c,e} p\left(\begin{smallmatrix}\texttt{Abc}\\ \texttt{DeF}\end{smallmatrix}\right)}, \forall q \in Q \tag{6.14}$$

$$\beta\left(q, \begin{smallmatrix}\texttt{Ann}\\ \texttt{DnF}\end{smallmatrix}, 0\right) = \frac{\sum_{b\neq D,c,e} p\left(\begin{smallmatrix}\texttt{Abc}\\ \texttt{DeF}\end{smallmatrix}\right)}{\sum_{b,c,e} p\left(\begin{smallmatrix}\texttt{Abc}\\ \texttt{DeF}\end{smallmatrix}\right)}, \forall q \in Q \tag{6.15}$$

  where b, c, e are ambiguous nucleotide symbols, $b, e \in \{A, C, G, T\}$ and $c \in \{G, R, Y, H, N\}$, such that `ADc`, `Abc` and `DeF` instantiate possible ambiguous codons with the fixed symbols `A`, `D`, `F` respectively.

This model generates words over the alphabet $\mathcal{A} = \{0, 1\}$, where 0 denotes mismatches and 1 denotes matches of amino acids with a +1 frameshift, defined as in Definition 13.

Background probabilities

The background model of random frameshifted alignments does not require any correlation between successive pairs of aligned symbols. In consequence, it is a simple Bernoulli model, independently generating random matches and mismatches with the probabilities

$$P(1) = \sum_{\texttt{ABC},\texttt{DEF}:\texttt{B=D}} p(\texttt{ABC}) \cdot p(\texttt{DEF}) \tag{6.16}$$

$$P(0) = 1 - P(1). \tag{6.17}$$

Relations (6.16) and (6.17) hold for +1 and +2 frameshifted alignments. The reasoning for other cases is very similar and therefore omitted here.

The models presented in this section can be integrated in IEDERA in order to determine the sensitivity of spaced seeds with respect to target frameshifted alignments. The actual design of such seeds is work in progress. In practice, a single contiguous seed of size k between 4 and 6 is used for heuristic similarity search.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 7

Experiments and discussion

7.1 Implementation

This part of the thesis focused on novel methods for detecting hidden protein similarities. First, a dynamic programming algorithm is proposed for aligning protein sequences with frameshifts, by back-translating the proteins into graphs that implicitly contain all the putative DNA sequences, using a scoring system taylored for this problem. Second, a heuristic approach based on seeding was designed for high speed similarity search in large databases. These concepts are implemeted in the Java software tool called PATH, currently available for download or via a web interface at `http://bioinfo.lifl.fr/path/`. The files containing translation dependent score matrices computed for several evolutionary distances are also available for download at the same URL.

7.2 Experimental results

We will further present several significant frameshifted alignments obtained with our method. The experimental results presented here were obtained in the following experimental setup: a search for frameshifted forward alignments was launched on samples from the full NCBI protein databases for several species, using a 00.50 base per codon divergence scoring matrix; the alignments with an E-value $< 10^{-9}$, presenting at least one significant frameshift, were selected.

***Yersinia pestis*: Frameshifted transposases** Figure 7.1 displays the alignment of two transposase variants from *Yersinia pestis*. Both proteins are widely present on the NCBI `nr` database. The mechanism involved is (most probably) a programmed translational frameshifting since such mechanism has been quite frequently observed in several other transposases from related species, e.g. as in E. coli [127].

***Xylella fastidiosa*: Frameshifted β-glucosidases** Two β-glucosidase variants from *Xylella fastidiosa* are aligned on Figure 7.2 with both variants widely present on the NCBI `nr` database. *Xylella fastidiosa* is a plant pathogen transmitted by *Cicadellidae* insects (*Homalodisca vitripennis*,*Homalodisca liturata*) and responsible for *phoney disease* on the peach tree, *leaf scorch* on the oleander, and *Pierce's disease* on grape. The β-glucosidase is usually required by several organisms to consume cellulose.

Interestingly, β-glucosidase frameshifts have already been studied in [175] on several bacteria including such γ-proteobacteria as *Erwinia herbicola* and *Escherichia coli* of *Enterobacteriales*

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

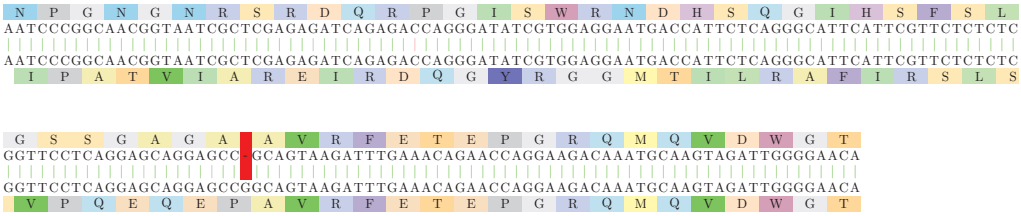

Figure 7.1: Yersinia pestis transposases: The alignment of two transposase variants from *Yersinia pestis*, [GenBank:167423046] – subsequence 4-167 of the back-translation, and [GenBank:EDR63673.1] – subsequence 225-389 of the back-translation. The frameshift mutation at position 115/336 corrects the reading frame. The frameshifted alignment fragment has an E-value of 10^{-7}.

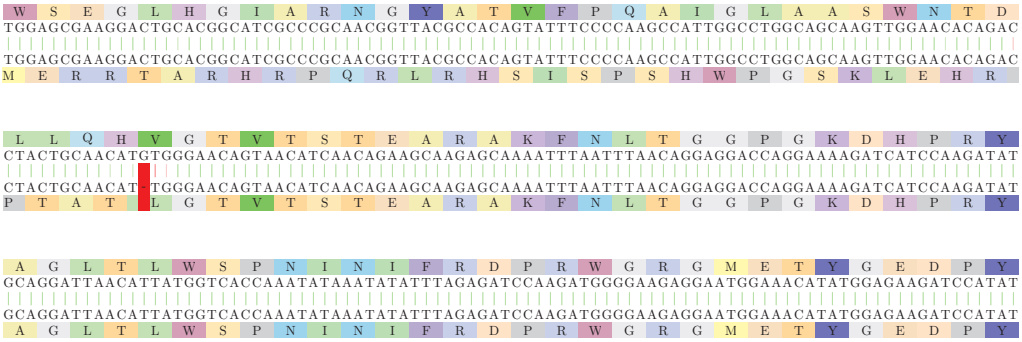

Figure 7.2: Xylella fastidiosa glucosidases: Two β-glucosidase variants from *Xylella fastidiosa*, [GenBank:AAO29662.1] – subsequence 202-2645 of the back-translation, and [GenBank:EAO32640.1] – subsequence 2-2444 of the back-translation. We only show in this image a fragment of the full alignment (the first 239 base pairs). The second part is not particularly interesting in our context because the sequences are aligned on the same reading frame, with a very small number of mismatches. In the first part, the sequences are aligned with a reading frame difference that is corrected starting with positions 304/104. The frameshifted alignment fragment has an E-value of 10^{-8}.

but not directly observed in *Xanthomonadales*.

Venom neurotoxins Diversification of venom toxins has been studied in [64] for advanced snakes: frameshifts were one of the most significant mechanisms involved in the "evolution of the arsenal" and its diversification toward specialized prey capture, sometime with a loss of neurotoxicity [124]. We thus studied neurotoxins from several higher snakes.

In Figure 7.3, we show the alignment of two presynaptic neurotoxins from two higher snakes of the Elapidae family (*Bungarus candidus* and *Naja kaouthia*). Most of the sites are conserved: the primary metal binding site and the putative hydrophobic channel remain before the frameshift, and only the fourth (and last) part of the catalytic network seems changed. We also noticed that, in the original second sequence, the Cysteine regions are more conserved at the DNA level than other amino acids, even after the frameshift, which is a strong hint of the non randomness of this part of the alignment.

Following the discovered frameshift of Figure 7.3, we took into consideration the sequences of *Bungarus candidus* species that were similar to the non-frameshifted presynaptic neurotoxin of *Naja kaouthia*. An interesting alignment is presented in Figure 7.4, showing a protein that aligns to it well but not perfectly (at least 4 non synonymous transitions before the frameshift and 1 transversion after): this lets open the potential "duplicated first then frameshifted" origin

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Figure 7.3: Elapidae neurotoxins (1): Two presynaptic neurotoxins from two higher snakes of the Elapidae family (*Bungarus candidus* and *Naja kaouthia*), [Swiss-Prot:Q8AY47.1] – subsequence 64-407 of the back-translation, and [PIR:PSNJ2K] – subsequence 13-354 of the back-translation. The sequences are aligned on the same reading frame up to position 186/135, and on a +1 reading frame from that point forward. The frameshifted fragment has an E-value of 10^{-9}.

of the frameshifted protein. This assumption was strongly supported by the alignment of the two corresponding cDNA [GenBank:AY057881.1] and [GenBank:AY057880.1] of two homologous proteins.

Furthermore, Figure 7.5 displays an alignment of two presynaptic neurotoxins from two higher snakes of the Elapidae family (*Laticauda colubrina* and *Laticauda laticaudata*). It shows that the unidentified peptide is in fact an alternative splicing (or frameshifted) variant of the neurotoxin. Note that BLASTP identification of the frameshifted peptide on the NCBI `nr` database gives high E-values (minimal is of 0.67) and is thus expected to be missed by automatic prediction tools (since most of the features specific to neurotoxin are not present), whereas the non frameshifted peptide fragment, once identified, gives several E-values $< 10^{-10}$ on NCBI `nr`.

Platelet-derived growth factor Platelet-derived growth factor is a potent mitogen for cells of mesenchymal origin. Binding of this growth factor to its affinity receptor elicits a variety of cellular responses. It is released by platelets upon wounding and plays an important role in stimulating adjacent cells to grow and thereby heals the wound. In Figure 7.6, we show the alignment of the back-translated *human* and *rat* platelet-derived growth factor proteins. The two proteins share high similarity at the amino acid level on the subsequences 1-84 and 113-195. The amino acids 85-112 can be easily aligned with a frameshift, as can be seen in Figure 7.6, while classic protein alignment reveals little similarity in these areas (Figure 7.7).

It is interesting to notice that this double frameshift (if confirmed) may have little influence on the protein (only the beginning of the receptor binding interface is modified). It is also interesting to notice that both the "inducing" and "correcting" frameshifts are located on two different exons.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

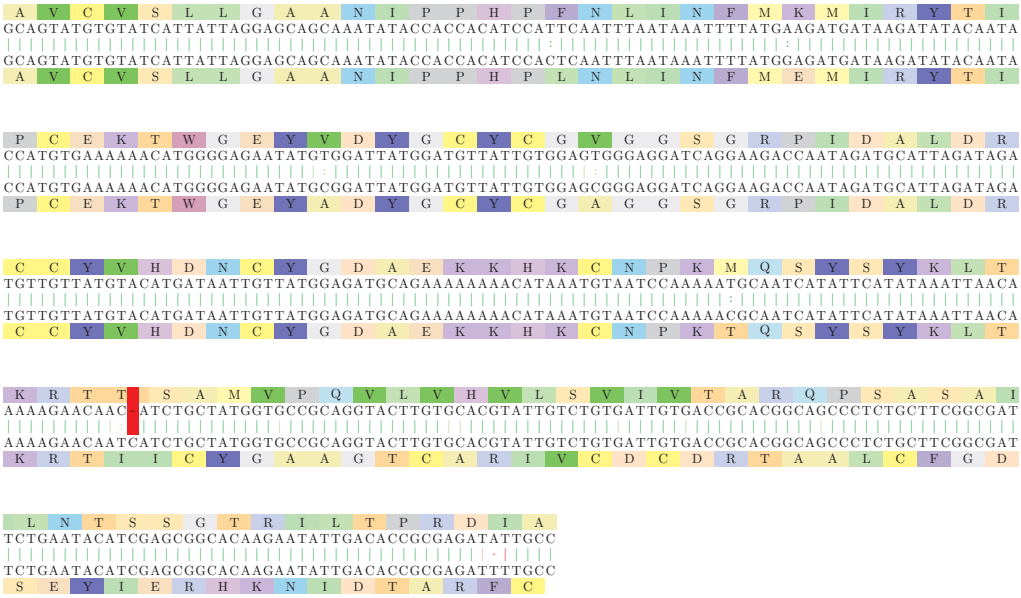

Figure 7.4: Elapidae neurotoxins (2): Two *Bungarus candidus* proteins, very similar at the DNA level ([Swiss-Prot:Q8AY47.1] and [Swiss-Prot:Q8AY48.1]). From the first 94 amino acid pairs, only 4 present mismatches (which are transitions at the coding DNA level). A frameshift mutation is visible at position 284 of the back-translated sequences. The fragments following it are almost perfectly aligned with a frameshift, with an E-value of 10^{-9}.

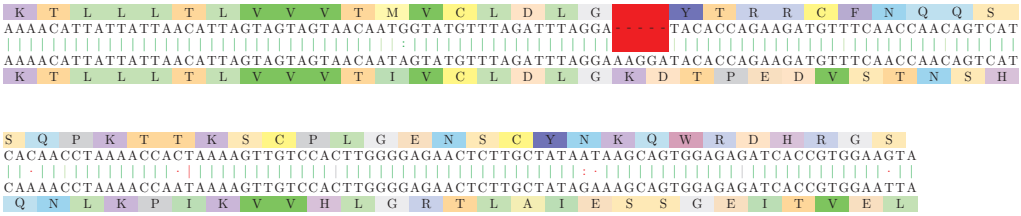

Figure 7.5: Elapidae neurotoxins (3): Two presynaptic neurotoxins from two higher snakes of the Elapidae family ([DDBJ:BAA75760.1] of *Laticauda colubrina* and [DDBJ:BAC78208.1] of *Laticauda laticaudata*): It shows that the unidentified peptide is in fact an alternative splicing (or frameshifted) variant of the neurotoxin. The frameshifted fragment has an E-value of 10^{-10}.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

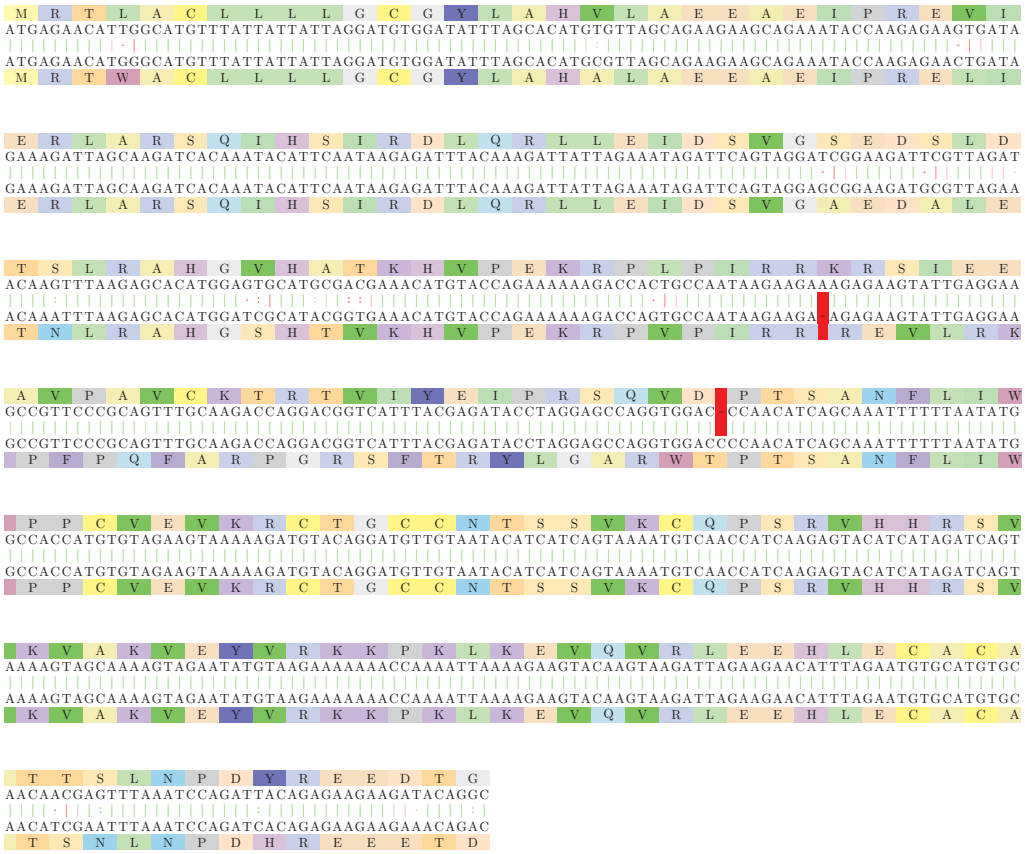

Figure 7.6: Platelet-derived growth factor proteins: The alignment of the back-translated platelet-derived growth factor proteins from *Homo sapiens* and *Ratus sp* ([Swiss-Prot:P04085.1] and [DDBJ:BAA00987.1]). The two proteins share high similarity at the amino acid level on the subsequences 1-84 and 113-195. The amino acids 85-112 can be easily aligned with a frameshift, with an E-value of 10^{-6}. Both the "inducing" and "correcting" frameshifts are located on two different exons.

```
PDGFA_HUMAN     1 MRTLACLLLLGCGYLAHVLAEEAEIPREVIERLARSQIHSIRDLQRLLEI   50
                  |||.|||||||||||||.||||||||||:|||||||||||||||||||||
BAA00987.1      1 MRTWACLLLLGCGYLAHALAEEAEIPRELIERLARSQIHSIRDLQRLLEI   50

PDGFA_HUMAN    51 DSVGSEDSLDTSLRAHGVHATKHVPEKRPLPIRRKRSIEEAVPAVCKTRT  100
                  ||||:||:|:|:|||||.|..||||||||:||||:..:.:..:|...:...
BAA00987.1     51 DSVGAEDALETNLRAHGSHTVKHVPEKRPVPIRRREVLRKPFPQFARPGR  100

PDGFA_HUMAN   101 VIYEIPRSQVDPTSANFLIWPPCVEVKRCTGCCNTSSVKCQPSRVHHRSV  150
                  .......::..||||||||||||||||||||||||||||||||||||||||
BAA00987.1    101 SFTRYLGARWTPTSANFLIWPPCVEVKRCTGCCNTSSVKCQPSRVHHRSV  150

PDGFA_HUMAN   151 KVAKVEYVRKKPKLKEVQVRLEEHLECACATTSLNPDYREEDT          193
                  ||||||||||||||||||||||||||||||::||||:|||:|
BAA00987.1    151 KVAKVEYVRKKPKLKEVQVRLEEHLECACATSNLNPDHREEET          193
```

Figure 7.7: Classic protein alignment of the platelet-derived growth factor proteins: The classic protein alignment of the platelet-derived growth factor proteins from *Homo sapiens* and *Ratus sp* ([Swiss-Prot:P04085.1] and [DDBJ:BAA00987.1]) shows very little amino acid similarity between the 85-112 subsequences, that we have successfully aligned on a +1 frameshift.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

7.3 Conclusions and perspectives

The problem of finding distant protein homologies, in particular affected by frameshift events, is addressed in this work from a codon evolution perspective. We search for protein common origins by implicitly aligning all their putative coding DNA sequences, stored in efficient data structures called back-translation graphs. The approach relies on a dynamic programming alignment algorithm for these graphs, which involves a non-monotonic gap penalty that handling frameshifts and full codon indels differently. We designed a powerful translation-dependent scoring function for nucleotide pairs, based on codon substitution models whose purpose is to reflect the expected dynamics of coding DNA sequences.

The approach is illustrated in this chapter with several alignment examples of known and hypothetical frameshifted proteins, some of which are not detectable via classic alignment methods because of the low coding sequence similarity. Such examples support our method's applicability in the discovery of distant protein homologies and functional frameshifts.

Future work regards further improvements and refinements of the scoring system, for example in order to focus on the detection of short double correcting frameshifts (two frameshifts separated by a small number of bases, where the second corrects the reading frame disrupted by the first). Such cases have been shown to occur frequently [175], but are often highly penalized by sequence comparison methods, that discard the correct alignment in favor of an ungapped one with a higher score.

Some extensions of this work include the support for multiple alignments of back-translation graphs. This feature can be useful for confirming frameshifts by similarity of the frameshifted subsequence with the corresponding fragments from several members of a family. Also, for boosting the efficiency, seeding techniques for back-translation graphs could be further analysed and explored.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Published papers and talks

Talk (2010):

Gîrdea, M. (2010). Back-translation for discovering distant protein homologies in the presence of frameshift mutations. *EuroDocInfo 2010*, January 21-22, Valenciennes, France.

Journal article (2010):

Gîrdea, M., Noé, L., and Kucherov, G. (2010). Back-translation for discovering distant protein homologies in the presence of frameshift mutations. *Algorithms for Molecular Biology*, 5(6), January 2010.

Conference paper (2009):

Gîrdea, M., Noé, L., and Kucherov, G. (2009). Back-translation for discovering distant protein homologies. In Salzberg, S. and Warnow, T., editors, *Proceedings of the 9th International Workshop in Algorithms in Bioinformatics (WABI), Philadelphia (USA)*, volume 5724 of *Lecture Notes in Computer Science*, pages 108–120. Springer Verlag.

Poster & talk (2008):

Gîrdea, M., Kucherov, G., and Noé, L. (2008). Protein sequence alignment via anti-translation. *JOBIM 08*, pages 157–158.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Part III

Algorithms for mapping SOLiD ™ reads

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 8

Context and motivation

The advent of high-throughput sequencing technologies, discussed earlier in Chapter 2, constituted a major advance in genomic studies, offering new prospects in a wide range of applications. Among these, the SOLiD system, presented in Section 2.2.3, features a 2-base encoding with an error-correcting capability helping to better distinguish between sequencing errors and biological SNPs. The platform produces short reads of 35-50 base pairs, represented as sequences of 4 colors prefixed by one base symbol which allows decoding the read unambiguously. These reads are generally mapped onto a reference genomic sequence in order to determine possible sequence variations such as SNPs and indels. The encoding uses 4 colors to denote the 16 possible pairs of adjacent nucleotides that can appear in a sequence. Because of the dually encoded nucleotide information in adjacent colors, read positions cannot be interpreted independently, which prevents classic pairwise alignment algorithms from capturing the DNA sequence similarities by simple color sequence alignment.

8.1 Properties of the 2-base SOLiD encoding

The theoretical principles behind the error-correcting properties of the SOLiD encoding are explained in [1]. A read is represented as an initial base followed by a sequence of overlapping dimers, each encoded with one of four colors using a degenerate coding scheme (Figure 8.1). In the following, $color(B_1B_2)$ will denote the color associated to the dinucleotide B_1B_2. The

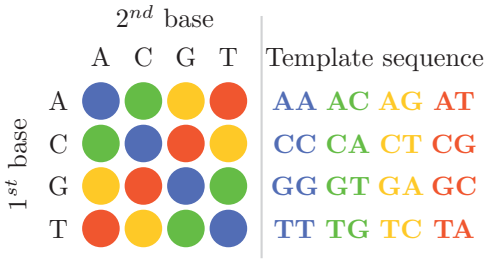

Figure 8.1: 2-base color encoding: (c1) a pair of bases has the same color regardless of the order; (c2) each base yields a different color when paired with a different base; (c3) all pairs of identical bases have the same color; (c4) a pair of adjacent bases has the same color as its complementary.

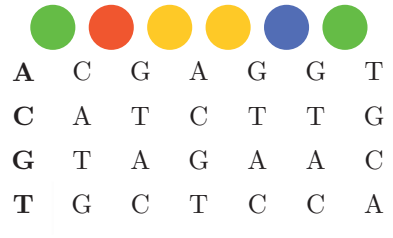

Figure 8.2: Unambiguous translation of a color sequence starting from a known base. According to rule **c2** (Figure 8.1), if the color and the first symbol of a dinucleotide are known, then the base second symbol can be uniquely determined.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

encoding satisfies several rules:

(c1) A pair of bases has the same color regardless of the order, e.g.,

$$color(AG) = color(GA)$$

(c2) Given a base, it always results in a different color when paired with a different base, e.g.,

$$color(\mathbf{A}T) \neq color(\mathbf{A}G) \neq color(\mathbf{A}C) \neq color(\mathbf{A}A)$$

and

$$color(T\mathbf{A}) \neq color(G\mathbf{A}) \neq color(C\mathbf{A}) \neq color(A\mathbf{A})$$

Hence, although a single color in a read can represent any of four dimers, the overlapping properties of the dimers and the nature of the color code eliminate ambiguities (Figure 8.2).

(c3) All dinucleotides formed with two identical bases have the same color:

$$color(AA) = color(CC) = color(GG) = color(TT)$$

(c4) A pair of bases and its complementary have the same color, e.g.

$$color(AG) = color(TC)$$

In conjunction with property **c1**, this allows obtaining the reverse complementary of a read by reversing the color sequence.

From Figure 8.1, we notice that:

- the color for two identical nucleotides (AA, CC, GG, TT) is blue (B);
- if both nucleotides are either purines (AG, GA) or pyrimidines (CT, TC) the associated color is yellow (Y);
- red (R) labels pairs of complementary bases (AT, TA, CG, GC);
- the other 4 dinucleotides (AC, CA, GT, TG), formed with one purine and one pyrimidine that are not complementary are colored green (G).

Since two different dinucleotides with the same base on the first position get different colors (rule **c2**), the nucleotide symbol on the first position and the color uniquely determine the nucleotide on their second position. We will denote by $next(B_1, color)$ the function that retrieves the second symbol in a dinucleotide, given the first symbol and the color associated to the pair. Obtaining the translation of a color sequence prefixed with a known base symbol is now straightforward: the nucleotide at position p is determined by the nucleotide at position $p-1$ and the $(p-1)$-th color in the sequence, as illustrated in Figure 8.2:

$$B_p = next(B_{p-1}, color_{p-1}) \tag{8.1}$$

In this context, colors can be seen as **transformations of bases** rather than dinucleotide labels. Let $B = \{A, C, G, T\}$ be the set of base symbols. A color c is a function $f_c : B \to B$ that "transforms" base B into base B'. Namely, f_{B} is the *identity function* since it maps each symbol to itself ($f_{\mathsf{B}}(A) = A$, $f_{\mathsf{B}}(C) = C$ etc), f_{Y} swaps a purine with another purine or a pyrimidine with another pyrimidine ($f_{\mathsf{Y}}(A) = G$, $f_{\mathsf{Y}}(C) = T$, etc), f_{R} maps each base to its complementary ($f_{\mathsf{R}}(A) = T$, $f_{\mathsf{R}}(C) = G$, etc), and f_{G} swaps A with C and G with T.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

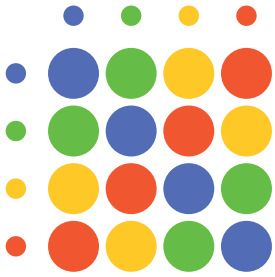

Figure 8.3: Color composition table, depicting the effect of two successive color transformations. Applying the same color twice successively on a base will lead to the same base symbol, thus having the same effect as the transformation by the color B (e.g. RR = B, GG = B). Similarly, we can determine that B in combination with any other color has the same effect as that color (e.g. YB = Y), and two colors other than B combined have the effect of the third color (e.g. RY = G).

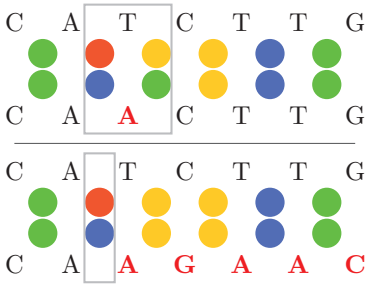

Figure 8.4: A simple rule for distinguishing SNPs form reading errors. Since each base is comprised in two consecutive colors, a SNP is visible in the color sequence as two consecutive color changes, where the new colors have the same composition as the original colors. In this illustration, RY = BG = G. Reading errors are usually isolated color changes which, if translated, would completely change the entire nucleotide sequence.

Entire sequences of colors can also be treated as transformations, and allow determining the base at an arbitrary position p without explicitly computing intermediate bases. For example, the result of applying the color sequence RG to the base T is C.

We can therefore define **function composition on colors**. The combined effect of applying two successive color transformations is illustrated in Figure 8.3. In this color composition table, lines are labeled with the first color, columns with the second, and each cell gives the single color transformation equivalent to the sequence.

The following observations can be made with respect to the color composition. First, each color composed with blue (B) is equivalent to itself alone, meaning that B has neutral effects in color composition, since it does not change the identity of the base it transforms (rule **c3**). Second, each color composed with itself has the same effect as B, due to the color transformation symmetry (rule **c1**). Third, any two different colors other than B have the combined effect of the third color (as a consequence of rules **c1** and **c2**). Additionally, we see that if the colors blue (B), green (G), yellow (Y) and red (R) are encoded by the binary string values 00 (0), 01 (1), 10 (2) and 11 (3) respectively, the color composition can be easily implemented as a bitwise XOR operation.

As a side observation, the set of four colors $C = \{\mathsf{B}, \mathsf{G}, \mathsf{Y}, \mathsf{R}\}$ along with the composition operation (denoted "$\oplus$") behaves like a Klein four-group [14], meaning that it has the following properties:

(g1) Closure: if $c_i \in C$ and $c_j \in C$ then $c_i \oplus c_j \in C$

(g2) Associativity: $(c_i \oplus c_j) \oplus c_k = c_i \oplus (c_j \oplus c_k)$

(g3) Identity: there is an element c_1 such that $c_1 \oplus c = c \oplus c_1 = c, \forall c \in C$ (the identity element is blue (B))

(g4) Inverse: $\forall c \in C\ \exists c^{-1} \in C$ such that $c \oplus c^{-1} = c^{-1} \oplus c = c_1$ (each color is its own inverse)

(g5) Commutativity: $c_i \oplus c_j = c_j \oplus c_i$

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

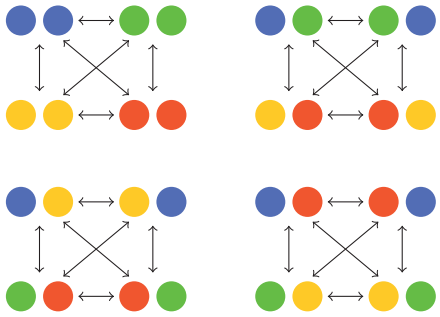

Figure 8.5: Detection of single base variations in color sequence alignments. One base change triggers two adjacent color modifications such that the modified pair has the same composition as the original pair. If the original colors are identical, then the modified colors must also be identical for the change to be valid. If the original colors are different, then valid changes are restricted to their reversal, or the other two colors regardless of the order.

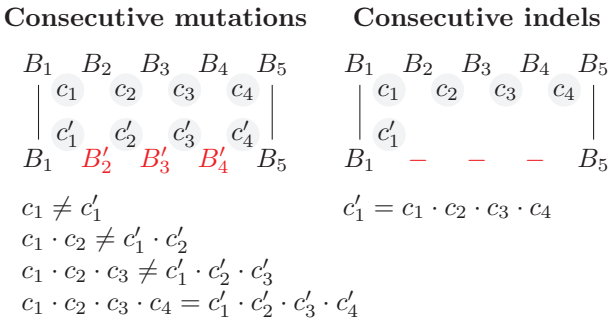

Figure 8.6: Detection of consecutive substitutions or indels in color sequence alignments. For **substitutions**, the composition of the color sequences that transform the same base B_1 into different bases (B_i, B'_i) must also be different (according to rule **c2**). While, in this setting, the first and last color in the two color subsequences corresponding to the series of substitutions always mismatch, there is no rule concerning the matches/mismatches of intermediary color pairs. In case of **indels**, all the colors encoding the bases that appear only in one sequence are replaced in the other by a single color equal to their composition, which encodes directly the dinucleotide formed by the remaining neighbors of the missing bases.

The colors as composable transformations are the foundation for deriving **rules that help the detection of nucleotide changes by color sequence comparison**. In the following, we will consider examples of local alignments of nucleotide sequences, with substitutions, insertions and deletions, and determine how these variations are reflected in the alignment of the corresponding color sequences.

In a correct local alignment, we expect any sequence mismatches to be flanked by significant zones of matching symbol pairs. If we focus on a single contiguous succession of mismatches in such an alignment, this basically means that, starting from a symbol B and passing through two different sequences of intermediary symbols, the same symbol B' is eventually reached. Considering the color code transformation and composition properties discussed above, this means that, in the respective color sequences, the corresponding colors will have equal compositions.

For example, **a single base change** is reflected in the color sequence by the change of two adjacent colors, corresponding to the two dinucleotides that contain the modified base (Figure 8.4). Note that a single color modification would completely change the translated nucleotide sequence. Here is a first proof of the error-correcting capabilities of the encoding. While with classic reads encoded directly as nucleotide sequences it can be difficult, under certain circumstances, to decide whether a mismatch is due to reading errors or is actually a mutation, the fact that in this 2-base encoding each position is practically interrogated twice helps to confirm detected changes. Not only must there be two adjacent colors that confirm the modification, but the valid color changes must satisfy the equal composition rule mentioned above (Figure 8.5).

This reasoning can be extended for **multiple consecutive base changes**, as illustrated in Figure 8.6 and Figure 8.7.

In the case of **consecutive substitutions**, the essential rule is that all intermediate bases are different in the two nucleotide sequences, meaning that partial color compositions starting

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

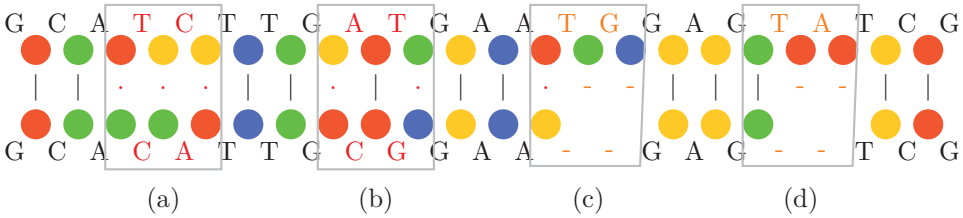

Figure 8.7: Illustration of various cases of consecutive substitutions and indels, and how they are reflected in the color mismatches: **(a)** 2 consecutive nucleotide substitutions, corresponding to 3 consecutive color substitutions in the color sequences; **(b)** 2 consecutive nucleotide substitutions, corresponding to a *mismatch - match - mismatch* sequence at the color level; this example illustrates the fact that only the first and last color in the subsequences corresponding to the variation must differ (according to rule **c1**, switching from the same base to two different bases results in two different colors and symmetrically, switching to the same base from two different bases results in two different colors), but there is no rule concerning the matches/mismatches of individual intermediary color pairs; here, the dinucleotide AT has the same color as the dinucleotide CG, resulting, at the color sequence level, in 2 apparently isolated color mismatches that are in fact part of subsequences with equal compositions and stand for a series of consecutive base substitutions; **(c)** 2 nucleotide deletions, reflected in the color sequences by one color substitution followed by 2 color deletions; the color Y preceding the gap encodes the dinucleotide AG formed by the remaining neighbors of the missing bases T and G; thus, color Y (in the second sequence) replaces colors R, G, B (in the first sequence), and $\mathsf{R} \oplus \mathsf{G} \oplus \mathsf{B} = \mathsf{Y}$; **(d)** 2 nucleotide deletions, reflected in the color sequences by 2 color deletions; incidentally, in this case there is no color substitution before the gap because the nucleotide T following the removed bases TA is identical to the first deleted nucleotide, and thereby produces the same color G in combination with its precedent G.

from the first color mismatch are also different in the color sequences (Figure 8.6). This does not necessarily imply that all intermediate color pairs mismatch, as depicted in Figure 8.7 **(a)** and **(b)**. Only the first and last color in the subsequences corresponding to the variation are necessarily different (according to rule **c1**, switching from the same base to two different bases results in two different colors and symmetrically, switching to the same base from two different bases results in two different colors). There is no rule concerning the matches/mismatches of individual intermediary color pairs. Hence, apparently isolated color mismatches that are part of subsequences with equal compositions may stand for a series of consecutive base substitutions, as shown in the example of Figure 8.7 **(b)**.

When **inserting or deleting bases**, all the colors corresponding to the bases that appear only in one sequence are replaced in the other sequence by a single color equal to their composition, which encodes directly the dinucleotide formed by the remaining neighbors of the missing bases (Figure 8.6, Figure 8.7 **(c)** and **(d)**). Occasionally, this single color might be identical to one of the colors in the other sequence, as illustrated in Figure 8.7 **(d)**.

In consequence, in alignment algorithms for 2-base color sequences, pairs of aligned colors should not be considered isolated. Instead, local color compositions are likely to provide the necessary information regarding variations in the corresponding nucleotide sequences.

8.2 Challenges of mapping SOLiD short reads

Some aspects that need to be taken into account when comparing color sequences follow directly from the nature of the encoding, and refer to recognizing SNPs and indels by analyzing several consecutive color positions simultaneously, as discussed in Section 8.1. In addition to these, read

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

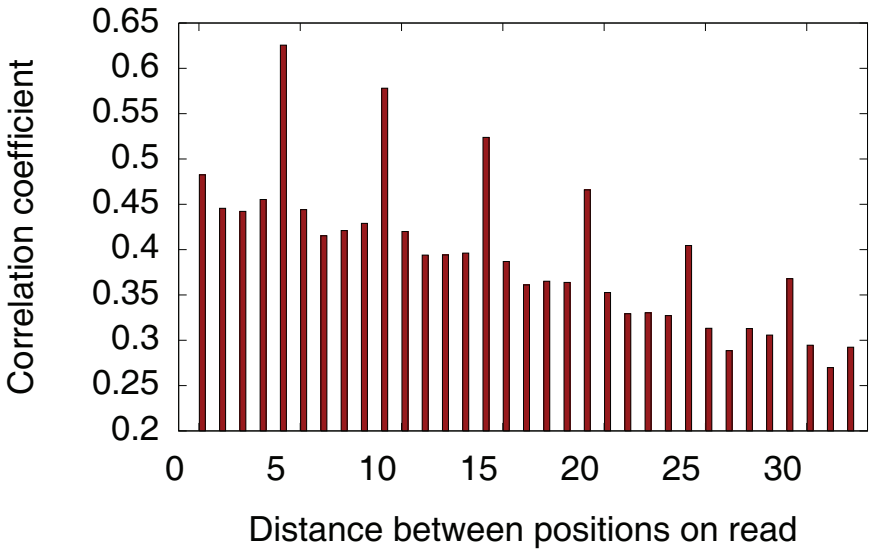

Figure 8.8: Position quality correlation coefficient depending on the distance between read positions.

mapping tools should take into consideration the possibility of reading errors. A good understanding of the most frequent error types and their distribution along the read can significantly improve the results.

In the SOLiD technology, the sequencing chemistry (presented earlier in Section 2.2.3) suggests periodical occurrences of reading errors. Basically, the sequencing by ligation process within the SOLiD platform relies on successive hybridizations of 8-mer oligonucleotides on the template to be sequenced. The oligonucleotides contain 3 universal base, 3 degenerate bases and 2 adjacent bases that identify two positions on the template, correlated with the identity of the fluorescent labels at their 5' end. After ligation, bases 6-8 are cleaved off, along with the fluorescent dye, leaving the 5' end available for another ligation. Hence, two positions p and $p+1$ are correctly base-paired after attaching one oligonucleotide, and the positions at distance 5 ($p+5$ and $p+6$) are determined by the next oligonucleotide. The nucleotides at positions that do not fit this pattern are determined in subsequent rounds. Five rounds consisting of several ligation cycles are necessary to cover the template. Therefore, we expect reading error biases to propagate during such a sequencing round, thus appearing with a periodicity of 5.

To confirm this intuition, we studied the variation of the reading error probability along the read by analyzing statistical properties of about a million of SOLiD reads of the *Saccharomyces cerevisiae* genome. In this analysis, we used the qualities Q_l associated to each position l on the read, which relate to the error probability p_e^l through $Q_l = -10 \cdot \log_{10}(p_e^l)$ [58].

We computed the quality correlation between read positions depending on the distance between them. Formally, if m is the read length, then for each $i \in \{1, .., m-1\}$, we computed the correlation through the following standard formula $c(i) = \frac{E((Q_j - \widetilde{Q})(Q_{j+i} - \widetilde{Q}))}{(\sigma_Q)^2}$, where $E(\cdot)$ is the expectation, $\widetilde{Q}$ the average quality along the read, and σ_Q the standard deviation of quality values. The result is given in Figure 8.8. It shows significantly higher correlations (up to 0.63) between pairs of positions located at distances that are multiples of 5. Additionally, we studied the behavior of reading error probability values along the read. As shown in Figure 8.9, the error probability tends to increase towards the end of the read, making the last positions of the color sequence less reliable when searching for similarities.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

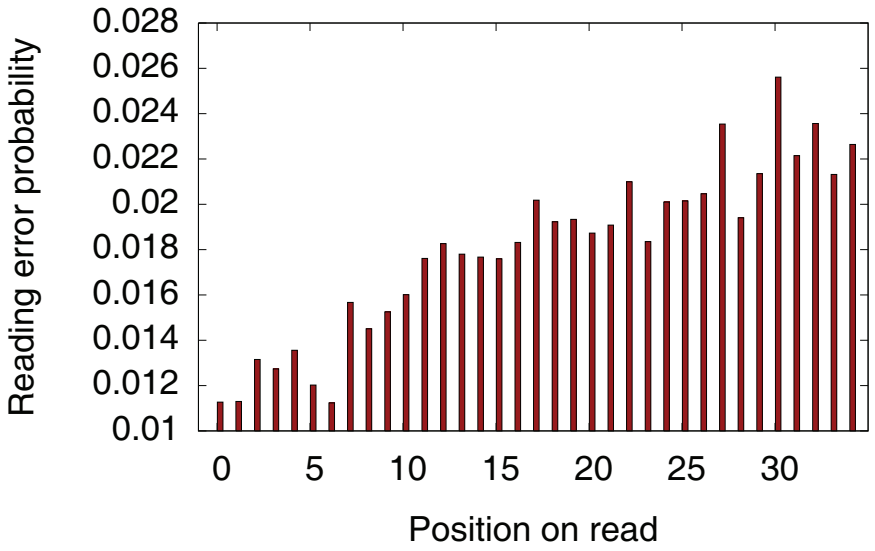

Figure 8.9: Average reading error probability at each read position.

These observations, in conjunction with the properties of the color encoding, serve as a basis for designing efficient and accurate methods for mapping SOLiD color-encoded reads to a reference genomic sequence. The goal is to achieve high sensitivity even in the presence of reading errors and SNPs/indels, and to provide a reliable method for assessing genomic variations and distinguishing them from misread data.

8.3 Related work

8.3.1 Read mapping software tools

A number of algorithms and associated software programs for read mapping have been recently published. Several of them such as MAQ [123], MOSAIK [195], MPSCAN [173] PASS [34], PerM [37], RazerS [208], SHRiMP [179] ZOOM [128], GASSST [174] or Crema [49] apply contiguous or spaced seeding techniques, requiring one or several hits per read. Other programs approach the problem differently, e.g., by using the Burrows-Wheeler transform (Bowtie [115], BWA [122], SOAP2 [126]), suffix arrays (segemehl [91], BFAST [92]), variations of the Rabin-Karp algorithm (SOCS [163]) or a non-deterministic automata matching algorithm on a keyword tree of the search strings (PatMaN [171]). Some tools, such as segemehl [91] or Eland [18], are designed for 454 and Illumina reads and thus do not deal with the characteristics of the SOLiD encoding. Also, it should be noted that, in many cases, sensitivity is sacrificed in favor of speed: most methods find similarities up to a small number of mismatches, and few approaches [91, 122, 171, 179] perform alignments between the read and the reference that account for nucleotide insertions and deletions.

8.3.2 Seeding techniques for color read mapping

Seed-based methods for read mapping use different seeding strategies. SHRiMP [179] uses spaced seeds that can hit at any position of the read and introduces a lower bound on the number of hits within one read. MAQ [123] uses six light-weight seeds allowed to hit in the initial part of

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

the read. ZOOM [128] proposes to use a small number (4-6) of spaced seeds each applying at a fixed position, to ensure a lossless search with respect to a given number of mismatches. Also in the lossless framework, PerM [37] proposes to use "periodic seeds" to save on the index size.

The approach encountered most often is to translate the reference sequence into colors and index it in the translated form. The spaced seed patterns used in practice usually present double "don't care"s in order to accept SNPs [37, 179], and some of the proposed seed families are guaranteed to be lossless for some combination of a small number of mismatches [37, 128].

A slightly different direction is taken in Crema [49], the difference consisting mostly in the hashing function associated to the spaced seed. For practical reasons, the reference is represented as a sequence of codes $\{0, 1, 2, 3\}$ corresponding to A, C, G, T respectively, while the reads are, in turn, sequences over the same set of integers $\{0, 1, 2, 3\}$ standing for the four colors of the SOLiD encoding B, G, Y, R. The authors introduce the *phase representation* of a read (color sequence) $c_1...c_m$, namely a sequence $\phi_0...\phi_m$, with $\phi_0 = 0$ and $\phi_k = \phi_{k-1} \oplus c_k, \forall k = 1..m$, where $\oplus$ is the XOR operation, as explained earlier in Section 8.1. Basically, $\phi_i = c_1 \oplus ... \oplus c_i$ is the composition of the entire color sequence up to position i, and, further composed with the code of the first nucleotide of the read t_0 would give the code t_i of the nucleotide at the ith position of the translation. More generally, if the code t_k at any position k of the translation is known, t_{k+j} can be obtained as $t_{k+j} = t_0 \oplus \phi_{k+j} = t_k \oplus \phi_k \oplus \phi_{k+j}$ since $t_0 = t_k \oplus \phi_k$ (following directly from the properties of the XOR operation $\oplus$: commutativity, associativity, $x \oplus x = 0$, $x \oplus 0 = 0 \oplus x = x$).

Consider a spaced seed π of length l and weight w, defined by the set of positions $\{\delta_1, ..., \delta_w\} \subseteq \{1, ..., l\}$. Applied at position i of some sequence $x_0...x_n$ (either nucleotide or color sequence), it selects the set of values $h(x_i...x_{i+l}) = y_1...y_w$ where $y_k = x_{i+\delta_1} \oplus x_{i+\delta_k}, \forall k = 1..w$. The seed is applied on the reference sequence $s_0...s_N$ and on the phased representation of the read sequence $\phi_0...\phi_m$, and a hit is found when $h(s_i...s_{i+l}) = h(\phi_j...\phi_{j+l})$, namely $s_{i+\delta_1} \oplus s_{i+\delta_k} = \phi_{j+\delta_1} \oplus \phi_{j+\delta_k}, \forall k = 1..w$. This relation is equivalent to $s_{i+\delta_k} = s_{i+\delta_1} \oplus \phi_{j+\delta_1} \oplus \phi_{j+\delta_k}, \forall k = 1..w$. In consequence, hits correspond to fragments of the read for which one of the four possible translations matches the reference according to the spaced seed pattern. Note that this approach does not handle reading errors inside the indexed fragment by design.

Despite the variety of proposed solutions, none of them relies on a *systematic method* taking into account (other than very empirically) *statistical properties of reads*, such as those discussed in Section 8.2. This aspect is adressed in this thesis in Chapter 9.

8.3.3 Alignment algorithms for color sequences

A straightforward approach to aligning reads (represented as color sequences) to a reference genome (a nucleotide sequence) is to translate either the reads or the genome into the alphabet of the other, which would allow performing the alignment with slightly adapted existing methods. Translating reads as nucleotide sequences is obviously a poor choice, since any reading error alters the translation of the entire nucleotide sequence following the misread position, as it was illustrated earlier in Figure 8.2, Section 8.1. Translating the genome into color-space is considered in some works [179] to lack rigor and prevent from properly distinguishing SNPs. Indeed, applying a *classic* alignment algorithm on color sequences does not allow to make this distinction. To cope with this issue, Chapter 10 proposes an algorithm for two color sequences

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

specifically designed for detecting reading errors, SNPs (isolated or consecutive), insertions and deletions.

Several alternatives to direct color sequence alignment have been proposed and implemented in existing read mapping tools.

The method adopted by SHRiMP [179] relies on the observation that, while a color-space error causes the rest of the sequence to be mistranslated, the genome will match one of the other three possible translations. The classical dynamic programming algorithm is thereby adapted to simultaneously align the genome to all four possible translations of the read, allowing the algorithm to move from one translation to another with the price of a sequencing error penalty. Basically, because the sequences are aligned in letter-space, the algorithm handles matches, mismatches, and indels, allowing the use of standard scoring schemes and affine gap penalties.

In a similar approach, the algorithm of [93] aligns a color sequence to a DNA sequence by exploring all possible nucleotide and color substitutions, in a 12-layer alignment matrix which can handle affine gaps.

In Crema [49], the *read alignment* problem is defined as the problem of aligning an unknown target sequence t to a known reference DNA sequence $s = s_{1..n}$, using a color sequence c that encodes t but may contain sequencing errors. In order to allow an evaluation with respect to the implied nucleotide mismatches and gaps, as well as the implied sequencing errors, an alignment is, in the context of [49], composed of columns contaning three cells: a *reference* cell s, a *color* cell c and a *target* cell t, the latter belonging to the inferred target sequence. All $s, c, t \in \{0, 1, 2, 3, \square\}$, with $\{0, 1, 2, 3\}$ being the numeric codes for $\{A, C, G, T\}$ (in the reference and the target sequences) and {B,G,Y,R} in the color sequence respectively, and $\square$ denoting indels. The columns describe seven types of alignment events: inferred nucleotide match with correct color (denoted M1 in the paper), inferred nucleotide match with color error (M2), inferred nucleotide mismatch with correct color (M3), inferred nucleotide mismatch with color error (M4), deletion from the reference sequence (D), insertion in the target sequence with correct color (I1), insertion in the target sequence with color error (I2). Indel characters may not occupy all three cells, and indels must appear together in the color and target cells.

Two algorithms for aligning a color sequence to a reference are proposed in [49]. First, a greedy algorithm obtains an alignment with at most d_{max} indels and at most e_{max} errors (nucleotide mismatches, color errors or indels, i.e. alignment colums different from M1). Second, a statistical alignment is performed on the basis of a pair-HMM which defines a probability distribution over alignments between nucleotide and color sequences. Basically, its states are of the form $\{M, I, D\} \times \{0, 1, 2, 3\}$, with M, I, D denoting events of match/mismatch, insertion and deletion, associated with a code $t \in \{0, 1, 2, 3\}$ of the last inferred nucleotide in the target sequence. Transitions emit pairs of symbols (s, c) from the reference and the color sequence respectively. The probability of generating erroneous colors is established within this model from the qualities associated to the read sequences, according to the classic relation $quality = \lfloor -10 \cdot \log_{10}(error\_probability) \rfloor$ which gives $error\_probability \approx 10^{-\frac{quality}{10}}$, and the rate of nucleotide substitutions between reference and target complies with an assumed Markov model of DNA sequence evolution. Finally, on this model, the best (most probable) alignment can be computed via dynamic programming [49].

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

8.4 Contributions

The following chapters focus on a rigorous and flexible algorithmic approach to mapping SOLiD color-space reads to a reference genome, capable to take into account various external parameters as well as intrinsic properties of reads resulting from the SOLiD technology.

As such, a first and most notable contribution, presented in Chapter 9, is a seed design framework based on Hidden Markov Models of read matches, using a formal finite automata-based approach previously developed in [110]. The main novelty of the method is an *advanced seed design* based on a *faithful probabilistic model of SOLiD read alignments* incorporating reading errors, SNPs and base indels, and, on the other hand, on a *new seeding principle* especially adapted for read mapping. The latter relies on the use of a small number of seeds *designed simultaneously with a set of position on the read where they can hit.* This principle of *positioned seeds* allows taking into account, in a subtle way, read properties such as a non-uniform distribution of reading errors along the read, or a tendency of reading errors to occur periodically at a distance of 5 positions, which are observed artifacts of the SOLiD technology. Both lossless and lossy spaced seeds can be designed in this framework, as will be explained in Chapter 9. For lossless seeds, where the goal is to detect all read occurrences within a specified number of mismatches, the approach offers the flexibility of partitioning this number into reading errors, SNPs and indels.

In addition to the advanced use of seed design, an alignment algorithm suited for color sequences is proposed in Chapter 10. For validating candidate mapping positions, a "base-intelligent" alignment method designed for color sequences makes advanced use of the encoding's error-correcting properties and implicitly takes into account the similarity at the DNA sequence level, thus improving the distinction between reading errors and SNPs.

Finally, these ideas are implemented in an experimental read mapping tool designed for SOLiD, discussed in Chapter 11 in comparison with several existing read mapping applications.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 9

Seed design for SOLiD reads

This chapter presents a methodology for designing *spaced seeds* to be used for mapping SOLiD color-space reads to a reference genome. By making use of their capacity to be adapted to different similarity search situations, this approach creates spaced seeds that capture statistical properties of sequences to be searched.

First, Section 9.1 proposes a seeding principle called *positioned seeds*, especially adapted for read mapping, consisting in the use of a small number of seeds *designed simultaneously with a set of position on the read where they can hit*. This setup suits read mapping tasks because the fixed lenght of the reads to be mapped and the known error distribution along the read allow gaining time by identifying and ignoring from the search process parts of the read containing unreliable information.

The design of both positioned and classic spaced seeds, lossless and lossy, is subsequently discussed in the following sections. In the lossy case we are allowed to miss a fraction of target matches, and the usual goal of seed design is to maximize the sensitivity over a class of seeds verifying a certain selectivity level. In the lossless case we must detect all matches within a given dissimilarity threshold (expressed in terms of a number of errors or a minimal score), and the goal of seed design is to compute a minimal set of seeds with the best selectivity that still ensures the lossless search. In the context of read mapping for high-throughput sequencing technologies, both lossy [123, 179] and lossless [37, 128] frameworks have been used.

The seed design approach proposed here relies on the methodology of [110], based on the finite automata theory. As explained earlier in Section 3.4.4, the central idea of the methodology idea is to model the set of target alignments by a *finite-state probability transducer*, which subsumes the Hidden Markov Model commonly used in biosequence analysis, and to represent a seed or a seed family, a *seed automaton* accepting the language of all alignments detected by that seed. Consequently, in this setup, it suffices to define an accurrate model of "good alignments" and integrate it into the Iedera software [110–112], which implements this methodology, in order to design sensitive or lossless spaced seeds for mapping SOLiD reads. Section 9.2 describes the such a model, that combines SNPs, indels and reading errors.

Section 9.3 focuses on lossless seed design, and proposes a fast algorithm for verifying the lossless property of a seed with respect to with a number of errors, with the possibility of partitioning them into SNPs, indels and reading errors.

Finally, Section 9.4 consists in a comparison and discussion of the seeds designed using the approaches described in this chapter.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

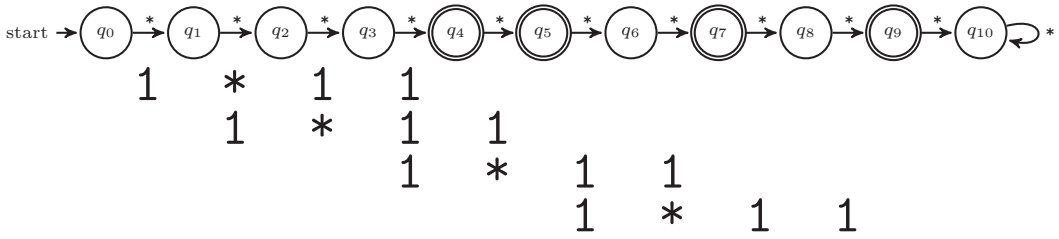

Figure 9.1: Example of position automaton λ_P for restricting the seed $\pi = 1*11$ to positions $P = \{0, 1, 3, 4\}$, on a read of length $m = 10$.

9.1 Positioned spaced seeds.

As shown Section 8.2, the reading error probability increases towards the end of the read, implying that a search for similarity within the last positions of the read could lead to erroneous results or no results at all. Hence, the seed selectivity can be improved by favoring hits at initial positions of the read where matches are more likely to be significant. We then define each seed π *jointly* with a set of positions P to which it is applied on the read.

We use the framework of [110] where a seed π is represented by a deterministic finite automaton $\mathcal{Q}$ over the alignment alphabet $\mathcal{A}$ which is here the binary match/mismatch alphabet. Let m be the read size. To take into account the set of allowed positions, we compute the product of $\mathcal{Q}$ with an automaton λ_P consisting of a linear chain of $m + 1$ states $q_0, q_1, \ldots, q_m$, where q_0 is the initial state, and for every q_i, both outgoing transitions lead to q_{i+1}. Final states of the automaton reflect the set of possible positions P where the seed is allowed to hit: a state q_i is final iff $i - s \in P$. A position automaton λ_P for restricting the seed $\pi = 1*11$ to positions $P = \{0, 1, 3, 4\}$, on a read of length $m = 10$ is illustrated in Figure 9.1.

The size of $\mathcal{Q}$ is a crucial parameter in the algorithm of [110] for computing the sensitivity of the seed. An efficient construction of such an automaton has been studied in [111]: it has the size at most $(w+1)2^{s-w}$ states, where s and w are respectively the *span* (length) and *weight* (number of *match* symbols) of the seed. A trivial upper bound on the size of the product automaton for a spaced seed of span s and weight w is $(w + 1) \cdot 2^{s-w} \cdot m$. This bound can be improved using the notion of matching prefix, as explained in [111]. Thus, an economical implementation of the product of $\mathcal{Q}$ by λ taking into account the set of matching positions P always produces at most $((w + 1) \cdot 2^{s-w} \cdot |P| + m)$ states.

9.2 Lossy framework

This section proposes a model for color sequence alignments, built on the properties of the SOLiD encoding presented earlier in Section 8.1 and on the observations of Section 8.2. Note that we consider the reference genome translated into the color alphabet, i.e. both the reads and the genome are represented in color space. This model of interesting/relevant alignments is integrated into the Iedera software [110–112], where the sensitivity of the designed seeds will be evaluated with respect to the alignments it represents.

As explained in Section 8.2, there are two independent sources of errors in reads with respect to the reference genome: reading errors and SNPs/indels, i.e., *bona fide* differences between the reference genome and sequenced data. We represent each of these sources by a separate Hidden Markov Model (viewed as a probabilistic transducer), combined in a model which allows all error

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

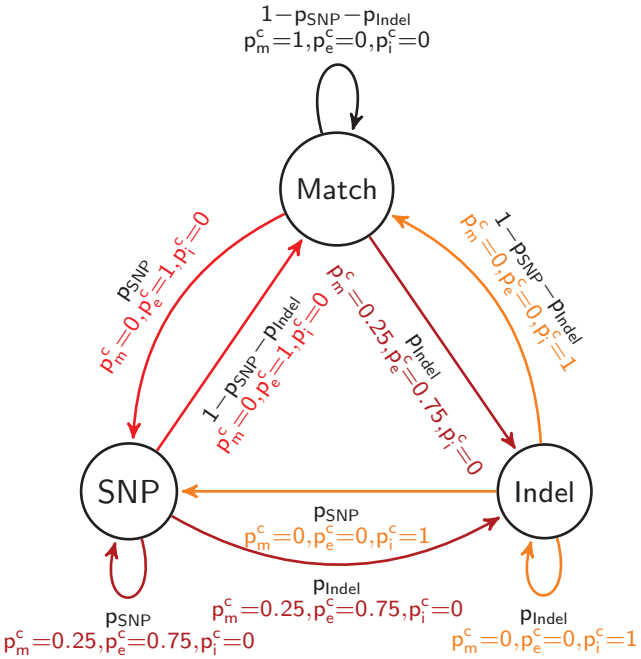

Figure 9.2: Model of SNPs and Indels ($M_{SNP/I}$). Colors of transitions correspond to emitted errors: black for color matches, red for mismatches, yellow for indels, and dark red for a mixture of matches (0.25) and mismatches (0.75).

types to be cumulated in the resulting sequences.

9.2.1 Biological variation HMM

The **SNP/Indel model**, denoted $M_{SNP/I}$, (Figure 9.2) has three states: *Match*, *SNP* and *Indel*, referring to matches, mismatches, and indels *at the nucleotide level*, and is parameterized by SNP and Indel occurrence probabilities, denoted p_{SNP} and p_{Indel}. Each transition of $M_{SNP/I}$ generates a *color match, mismatch or indel*, with probabilities p_m^c, p_e^c, and p_i^c respectively, defined as follows. An insertion or deletion of n nucleotides appears at the color level as an insertion/deletion of n colors preceded in 3/4 cases by a color mismatch [1]. Hence, the $p_e^c = 0.75$ for transitions incoming to the *Indel* state, and $p_i^c = 1$ for any transition outgoing from the *Indel* state. A nucleotide mutation is reflected in the color encoding by a change of two adjacent colors (and, more generally, n consecutive mutations affect $n+1$ consecutive colors [1]). Thus, $p_e^c = 1$ when entering or leaving the *SNP* state, and a color match/mismatch mixture when staying in the mismatch state, since color matches may occur inside stretches of consecutive SNPs. Finally, $p_m^c = 1$ when looping on the M state.

9.2.2 Reading error HMM

The **reading errors** are handled by a more complex model, denoted M_{RE} (Figure 9.3). Basically, it is composed of several submodels, one for each possible arrangement of reading errors on a cycle of 5 positions. Within these submodels, the transitions shown in red correspond to periodic reading errors, and generate reading errors with a fixed probability p_{err}. This simulates the periodicity property shown in Figure 8.8, Section 8.2. Switching from one cyclic submodel

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

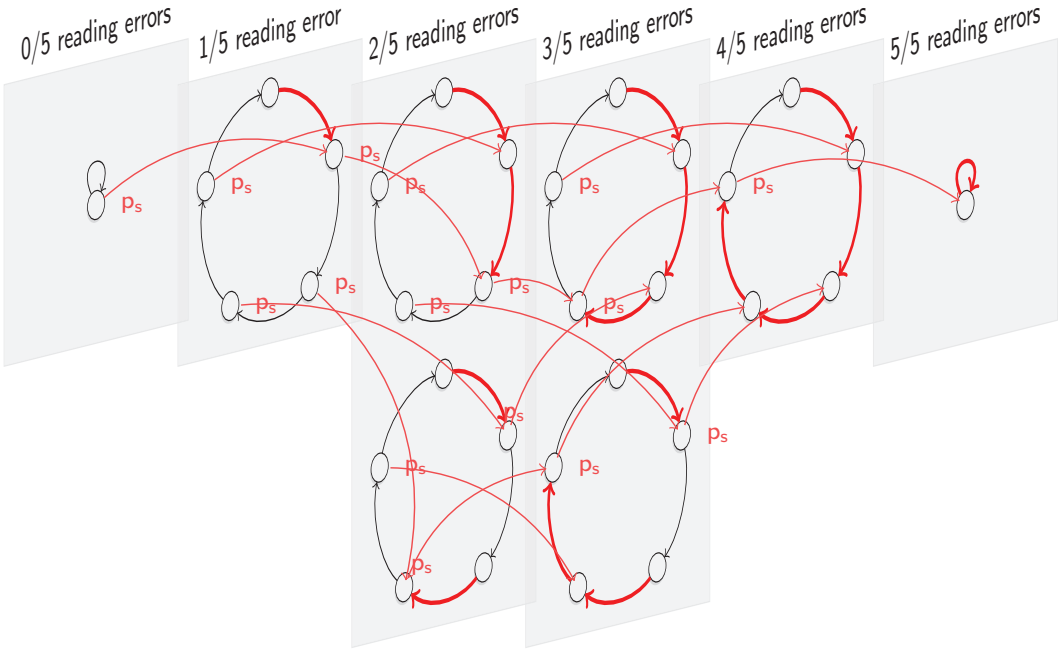

Figure 9.3: Reading error HMM (M_{RE}).

to another with a higher reading error rate (by adding another red transition with high error probability) can occur at any moment with a fixed probability p_s.

The transitions shown in black in the model in Figure 9.3 have an error emission probability of 0. However, in the complete reading error model, we wish to simulate the error probability that increases towards the end (in conformity with Figure 8.9). We do this by ensuring that reading errors are generated by these transitions with a probability $p'_{err}(pos)$ (lower than p_{err}) given by an increasing function of the current position *pos* on the read. Technically, this is achieved by multiplying the automaton in Figure 9.3 by a linear automaton with $m+1$ states, where m is the read length and the i-th transition generates a reading error (color mismatch) with the probability $p'_{err}(i)$. The reading error emission probability in the product model is computed as the maximum of the two reading error probabilities encountered in the multiplied models.

9.2.3 Combining biological variations and reading errors

The **final model**, which combines both error sources, is the product of $M_{SNP/I}$ and M_{RE}. While the states and transitions of the product model are defined in the classic manner, the emissions are defined through specific rules based on symbol priorities. If corresponding transitions of $M_{SNP/I}$ and M_{RE} generate symbols α and β with probabilities p_1 and p_2 respectively, then the product automaton generates the dominant symbol between α and β with probability $p_1 p_2$. Different probabilities obtained in this way for the same symbol are added up.

The dominance relation is defined as follows: *indels* are dominant over both *mismatches* and *matches*, and *mismatches* dominate *matches*. For example, $(indel, mismatch)$ results in an *indel*, $(mismatch, mismatch)$ and $(match, mismatch)$ represent *mismatch*, $(match, match)$ is a *match*. This approach ensures that errors generated by each of the two models are superposed, as illustrated in Figure 9.2.3.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

$$
\begin{array}{cl}
M_{SNP/I} & 1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,0\,\mathtt{I}\,1\,1\,1\,1\,1\,\mathtt{I}\,1\,1\,1\,1\,1\,1 \\
\times & \\
M_{RE} & 1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0 \\
= & \\
M_{(SNP/I)\times RE} & 1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1\,0\,\mathtt{I}\,1\,1\,1\,1\,0\,\mathtt{I}\,0\,1\,0\,1\,0\,0
\end{array}
$$

Figure 9.4: The model of alignments which combine both error sources, i.e. biological variations and reading errors, is **the product of** $M_{SNP/I}$ **and** M_{RE}, and generates alignments where all types of errors are commulated, as shown in the example above. Here, `1` denotes color matches, `0` represents color mismatches, and `I` stands for insertions in the read.

9.2.4 Seed sensitivity with respect to target alignments

The sensitivity of a seed family is defined [105,133] as the probability for at least one of the seeds to hit a read alignment with respect to a given probabilistic model of the alignment. As outlined earlier in Section 3.4.4, Chapter 3 this is done using the dynamic programming technique of [110] on the product of the automaton $\mathcal{Q}$ specifying a family of seeds possibly restricted to a set of positions, and the model of target alignments constructed as explained above.

9.3 Lossless framework

In the lossless framework, we have to test if the seed specified by $\mathcal{Q}$ is lossless, i.e. hits *all* the target alignments. The set of target alignments is defined through a threshold number of allowed mismatches.

9.3.1 Efficient algorithm for testing the lossless property

A straightforward way to test the lossless property of $\mathcal{Q}$ is to construct a deterministic automaton recognizing the set of all target alignments and then to test if the language of this automaton is included in the language of $\mathcal{Q}$. This, however, is unfeasible in practice. The automaton of all target alignments is much too costly to construct: for example, in the case of threshold of k mismatches, there are $\sum_{a=0}^{k}\binom{m}{a}$ different alignments of length m, and the Aho-Corasick automaton of these strings would have $\sum_{a=0}^{k+1}\binom{m}{a}$ states. Moreover, testing the inclusion would lead to computing the product of this automaton with $\mathcal{Q}$, which would multiply the number of states of this automaton by the number of states of $\mathcal{Q}$.

Alternatively, we propose an efficient dynamic programming algorithm directly applied to $\mathcal{Q}$ that can verify the inclusion (Algorithm 3). This algorithm computes, for each state q of $\mathcal{Q}$, and for each iteration $i \in [1..m]$, the minimal number of mismatches needed to reach q at step i. Let k be the threshold for the number of mismatches. Then, the lossless condition holds if and only if, at step m, all non-final states have a number of mismatches greater than k. Indeed, if there is a non-final state that has a number of errors at most k after m steps, then there is at least one string of length m with at most k mismatches that is not detected by the automaton, which contradicts the lossless condition. This algorithm is of time complexity $\mathcal{O}(|\mathcal{Q}| \cdot |\mathcal{A}| \cdot m)$, and space complexity $\mathcal{O}(|\mathcal{Q}| \cdot |\mathcal{A}|)$, where $\mathcal{A}$ is the alphabet of the alignment sequences.

To illustrate the efficiency of this algorithm, consider the case of a single spaced seed of span s and weight w, yielding an automaton with at most $(w+1) \cdot 2^{s-w}$ states [30, 110]. On this automaton, our method runs in time $O(m \cdot w2^{s-w})$ which brings an improvement by a factor of

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Input:

m = read length

$\mathcal{Q} = \langle S, \mathcal{A}, \delta, q_0, F \rangle$ the deterministic seed automaton (see also Definition 6, Section 3.4.4)

k = the threshold for the number of mismatches

Output: *true* if the seed represented by $\mathcal{Q}$ is lossless for alignments of length m with at most k mismatches; $false$ otherwise

begin

```
/* Initialisation                                                    */
foreach state q of Q do
    cost(q, 0) ← 0 ;
end
/* Mismatches have the cost 1                                        */
unit_cost[0] = 1;
/* Matches have the cost 0                                           */
unit_cost[1] = 0;

/* Recursion                                                         */
foreach iteration i ∈ [1..m] do
    foreach state q of Q do
        /* δ^{-1}(q, a) retrieves the set of states in Q that transition to
           state q with symbol a                                     */
        P ← δ^{-1}(q, a);
        /* Compute the minimal number of mismatches needed to reach q at
           step i                                                    */
        cost(q, i) ← min_{a∈A, q'∈P} cost(q', i − 1) + unit_cost[a];
    end
end

/* The lossless condition holds iff at step m, all non-final states have
   a number of mismatches greater than k.                            */
foreach non-final state q of Q do
    if cost(q, m) ≤ k then
        return false;
    end
end
return true;
end
```

Algorithm 3: Lossless property verification algorithm.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

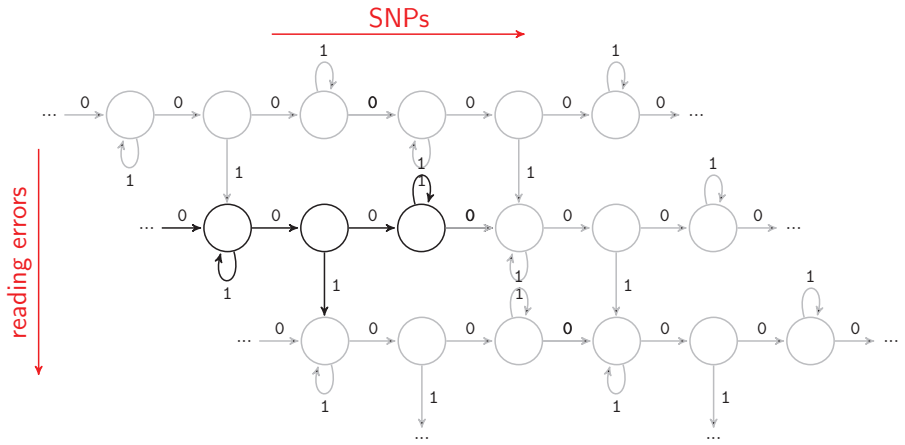

Figure 9.5: Building an automaton for k SNPs and h color mismatches from a repeated 3-state pattern.

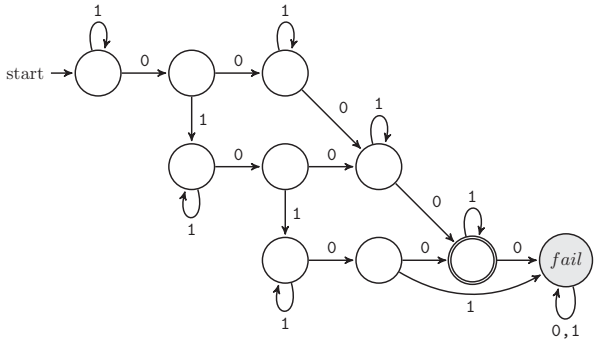

Figure 9.6: 1 SNP & 2 errors automaton.

$\frac{2^w}{w}$ of the general bound $\mathcal{O}(m2^s)$ from [32].

9.3.2 Lossless seeds with respect to SNPs and reading errors

In the context of color sequence mapping, it is interesting to define the lossless property with respect to a *maximal number of allowed mismatches that is split between SNPs and reading errors*. Since, in the color space, a SNP appears as two adjacent color mismatches, having k non-consecutive SNPs and h color mismatches implies the possibility to accept $2k+h$ mismatches with the additional restriction that there exist at least k pairs of adjacent ones. The automaton that recognizes the set of alignments verifying this condition on mismatches can be obtained by combining simple 3-state building blocks as depicted in Figure 9.5. An example of such an automaton, accepting 1 SNP and 2 reading errors, is illustrated in Figure 9.6 (`1` and `0` denote *match* and *mismatch* respectively).

Note that the case of consecutive SNPs, resulting in sequences of adjacent color mismatches, is a simpler problem (since consecutive SNPs produce less mismatches in the color representation than the same number of non-consecutive SNPs) and is covered by the proposed model: a seed that is lossless for alignments with non-consecutive SNPs will also be lossless for alignments with the same number of consecutive SNPs.

To verify the lossless property for k SNPs and h color mismatches, we intersect the corresponding automaton with the seed automaton (thus restricting the set of alignments recognized

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

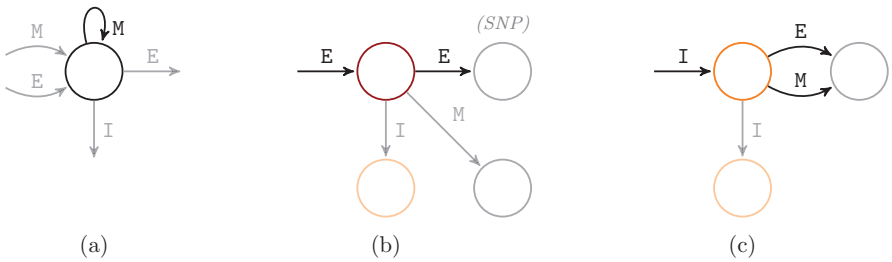

Figure 9.7: Modeling matches (a), reading errors, SNPs (b) and indels (c). If we consider the costs 0, 2, 3 and 4 respectively: (a) the cost of a match is 0; (b) the cost of single mismatch is 2, and a mismatch that follows it has the cost 1, summing to 3 which is the cost of a SNP; (c) the cost of an indel is 4, and a subsequent mismatch is accepted at zero additional cost.

by the seed to those with k SNPs and h color mismatches) and submit the result to the dynamic programming algorithm described above.

9.3.3 Lossless seeds with respect to SNPs, reading errors and indels

In a similar approach, a more complex automaton that takes into account insertions and deletions can be constructed, and intersected with the seed automaton in order to compute its lossless property using the given dynamic programming algorithm. We chose a cost for each event (*match*, *mismatch* – reading error or SNP, and *indel*), and with the same reasoning as above we consider alignments under a certain total cost rather than under a certain number of mismatches. In our experiments, we have chosen to use the costs: 0 for match, 2 for one color mismatch, 3 for one SNP (equivalent to having a 1 cost for a color mismatch preceded by another color mismatch), and 4 for indels.

In Section 9.2, we explained how modifications at the DNA sequence level are visible as differences between the associated color sequences. Following the same principle, we can build automata that represent alignments with a limited number of errors, in various combinations. The building blocks of such automata are represented in Figure 9.7. Sequences of matches are represented by looping into the same state without any cost modification (a). SNPs are represented by two consecutive color mismatches with different costs, while reading errors are represented as isolated mismatches (b). Finally, indels of bases correspond to indels of colors that may or may not be followed by a color mismatch (c), hence color mismatches preceded by an indel event are considered to have zero costs. A complete automaton example is not given here for complexity reasons. The next section will present seeds that are lossless for alignments with cost 7 under the cost associations given above, which accounts for several possible error combinations: 1 *indel* and 1 *SNP*, 1 *indel* and 1 *reading error*, 1 *SNP* and 2 *reading errors*, or 2 *SNPs*.

Note that our seeds are *not* indel seeds [135], i.e. the seed alphabet does not contain a symbol for insertions and deletions. Instead, the seeds must be placed *between* the indels within the alignments modeled by this automaton.

9.4 Designed seeds

We present now several efficient seed designs illustrating our methodology (more examples can be found at `http://bioinfo.lifl.fr/yass/iedera_solid`).

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

1-Lossy-10p: sensitivity 0.9543

```
1   5    10   15   20   25   30
1111111**111  :    :    :    :
: 1111111**111:    :    :    :
:   1111111**111   :    :    :
:   : 1111111**111 :    :    :
:   :   1111111**111    :    :
:   :    :1111111**111  :    :
:   :    :  1111111**111:    :
:   :    :    :1111111**111  :
:   :    :    :   1111111**111
:   :    :    :    :  1111111**111
```

2-Lossy-10p: sensitivity 0.9627

```
1   5    10   15   20   25   30
1111111111    :    :    :    :
:   : 1111111111   :    :    :
:   :    :  1111111111  :    :
:   :    :    :    1111111111:
:   :    :    :    :    1111111111
111111****1111:    :    :    :
:   : 111111****1111    :    :
:   :    :   111111****1111  :
:   :    :    :  111111****1111
:   :    :    :    :111111****1111
```

1-Lossy-12p: sensitivity 0.9626

```
1   5    10   15   20   25   30
1111111**111  :    :    :    :
: 1111111**111:    :    :    :
:   1111111**111   :    :    :
:   : 1111111**111 :    :    :
:   :   1111111**111    :    :
:   :    :1111111**111  :    :
:   :    :  1111111**111:    :
:   :    :    1111111**111   :
:   :    :    : 1111111**111 :
:   :    :    :   1111111**111
:   :    :    :    :1111111**111
:   :    :    :    :  1111111**111
```

2-Lossy-12p: sensitivity 0.9685

```
1   5    10   15   20   25   30
1111111111    :    :    :    :
:   : 1111111111   :    :    :
:   :    :1111111111    :    :
:   :    :    :1111111111    :
:   :    :    :    :1111111111
:   :    :    :    :    1111111111
1111**11**1111:    :    :    :
: 1111**11**1111   :    :    :
:   1111**11**1111 :    :    :
:   : 1111**11**1111    :    :
:   :   1111**11**1111  :    :
:   :    :    :    :1111**11**1111
```

Figure 9.8: Positioned seeds for 10 (above) and 12 (below) allowed positions. Different placements of a seed correspond to the allowed positions.

9.4.1 Lossy seeds

We first computed several sets of lossy seeds of weight 10, restricted to either 10 or 12 positions among the 34 positions of SOLiD reads, each including one or two seeds. Figure 9.8 shows some of the resulting seeds, together with the corresponding sensitivity values, computed through the methods described in Section 9.2.

Interestingly, both single seeds 1-Lossy-10p and 1-Lossy-12p contain a double gap, which may reflect that an SNP modifies two adjacent colors. However, this gap is not centered but rather shifted at the two-third of the seed (as observed for the best single seeds of [109]). Note also that in the two-seed families 2-Lossy-10p and 2-Lossy-12p, one of the chosen seeds is ungapped. This may be a consequence of the fact that we consider indels in our lossy model, which usually forces the seeds to have a smaller span. Another interesting observation is that two-seed families 2-Lossy-10p and 2-Lossy-12p are actually lossless for the threshold of 3 mismatches, whereas single seeds 1-Lossy-10p and 1-Lossy-12p are not lossless for this setting.

9.4.2 Lossless seeds

For SNPs and reading errors

We then focused on the lossless case where the maximal number of allowed mismatches is split between SNPs and reading errors. Using the procedure described in Section 9.3, we computed lossless single and double seeds for one SNP and two reading errors. Results are shown in Figure 9.9.

Note that the seed 1-Lossless-14p is one of several single seeds of weight 10 we found that satisfied this lossless condition, with no restriction on allowed positions. Interestingly, they all have a very large span (21) and a regular pattern with a periodic structure that can be obtained by iterating a simpler pattern solving the lossless problem for an appropriate *cyclic problem*,

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

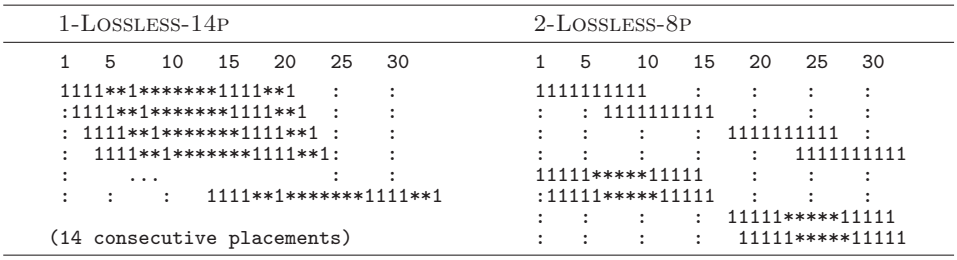

Figure 9.9: Lossless positioned seeds for 1 SNP and 2 reading errors.

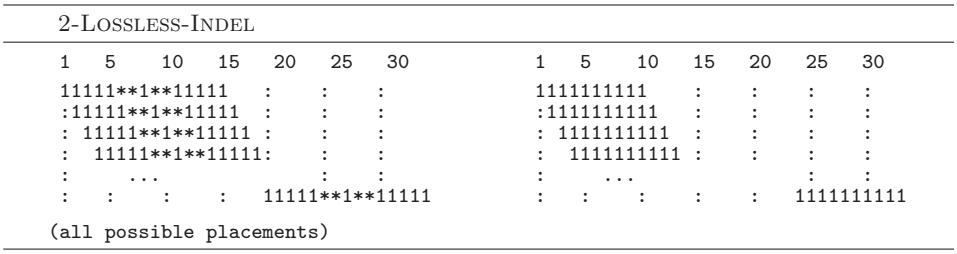

Figure 9.10: A family of two spaced seeds, lossless for 4 different error combinations: 1 indel and 1 SNP, 1 indel and 1 reading error, 2 SNPs, or 1 SNP and 2 reading errors.

following the property we previously described in [109]. For two-seed families, Figure 9.9 shows a lossless pair of seeds 2-Lossless-8p for read length 33 (which then remains lossless for larger lengths), where each seed is restricted to apply to four positions only.

For SNPs, reading errors and indels

Figure 9.10 displays a seed family that is lossless for 1 indel and 1 SNP or 1 indel and 1 reading error, 2 SNPs, or 1 SNP and 2 reading errors. As in the case of lossy seeds where indels are taken into account, these lossless seeds tend to have smaller lengths and therefore fewer gaps, in order to avoid any possible indel position.

9.4.3 Seed comparison

Figure 9.11 compares the theoretical selectivity/sensitivity of several single seeds, for weight ranging from 11 to 14, depending on the number of read positions the seed can be applied at. Note that restricting the number of possible hitting positions affects the seed template. The red polyline connects points corresponding to seeds optimized without restrictions on the number of positions. Relative to this line, we observe a good performance of seeds restricted to 24, 16 and 12 positions, with corresponding polylines lying above the red one, which means a better sensitivity/selectivity trade-off. This confirms that positioned seeds can be superior to unrestricted seeds, taking advantage of preferentially hitting positions where errors are less likely.

Furthermore, to get a better idea of the sensitivity of the obtained seeds applied to real data, we tested them on 100000 reads of length 34 from *Saccharomyces cerevisiae* and computed the number of read alignments hit by each (single or double) seed. Alignments were defined through the score varying from 28 to 34, under the scoring scheme +1 for match, 0 for color mismatch or SNP, -2 for gaps. Results are presented in Figure 9.12. One conclusion we can draw is that the

http://doc.univ-lille1.fr

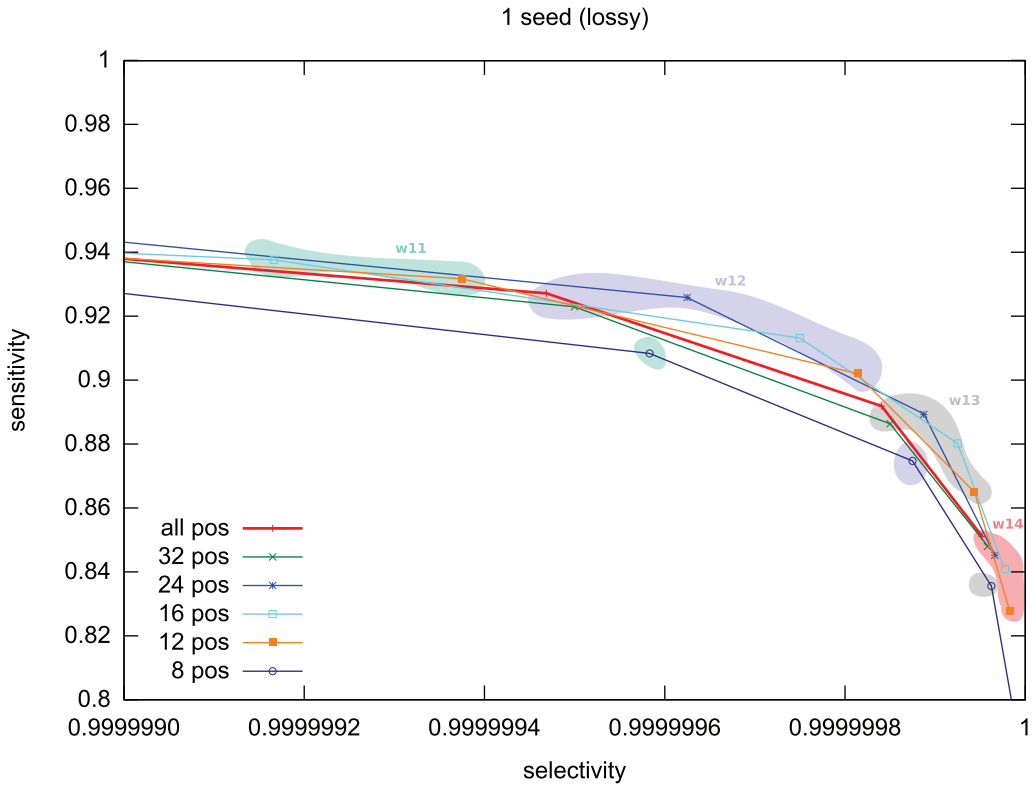

Figure 9.11: Theoretical selectivity/sensitivity of single seeds. X-axis corresponds to selectivity (1 minus the probability of hitting a random alignment) and Y axis to sensitivity. Color clouds highlight seeds of the same weight.

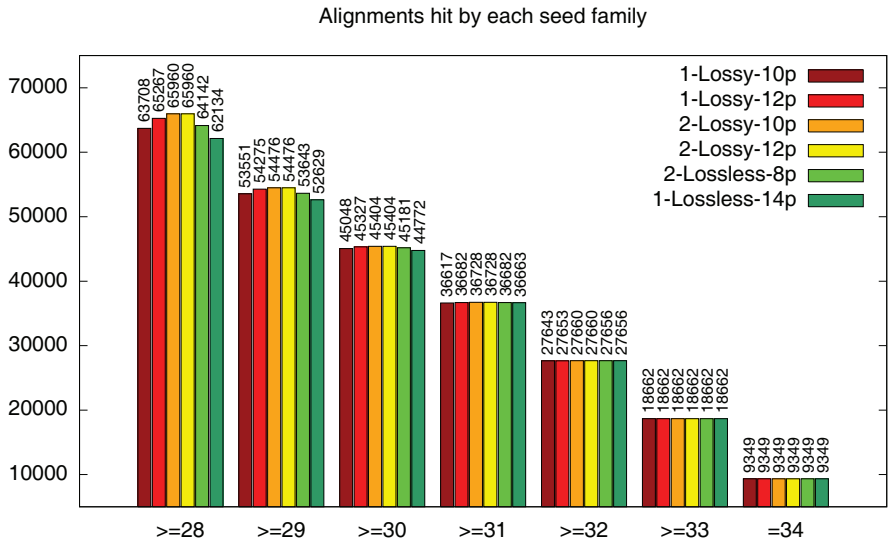

Figure 9.12: Number of read alignments with scores between 28 and 34 hit by each seed.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

performance of lossless seeds 1-Lossless-14p and 2-Lossless-8p decreases quite fast when the alignment score goes down, compared to lossy seeds. Intuitively, this is, in a sense, a price to pay for the lossless condition which usually makes these seeds less appropriate for the alignments with a number of errors exceeding the threshold. Another observation is that, as expected, single seeds perform worse than double seeds, although the overall number of positions where seeds apply is the same for both single and double seeds.

Note finally that the choice of the best seed can be affected, on the one hand, by different properties of the class of target alignments (number, type and distribution of mismatches and indels etc.) and, on the other hand, by the size of the data and available computational resources. The former can be captured by our probabilistic models described in Section 9.2. The latter is related to the choice of the selectivity level, directly affecting the speed of the search, which is defined by the seed weight and the number of allowed positions. Depending on the chosen selectivity, different seeds can (and should) be preferred. Note in this regard that seeds appearing in Figure 9.12 have different selectivity and are then incomparable *stricto sensu*.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 10

Color sequence alignment

The previous chapter focused on the design of spaced seed families for identifying candidate mapping positions for each read. The seed hits are *extended* using two alignment algorithms for color sequences, described below. The first is a fast SIMD bandwidth alignment algorithm of two color sequences which acts as a second filter, with the goal of eliminating most of the false positive hits of the spaced seeds. The alignments that pass this second filter are then submitted to the algorithm for aligning two color sequences of Section 10.3. The goal of this step is to obtain alignments of color sequences that are meaningful at the nucleotide level, i.e. to make the distinction between mismatches caused by reading errors and by SNPs and indels, and to properly assign the corresponding score penalties. The best scoring candidate mapping positions are stored for each read.

With the purpose of being an instrument for evaluating the significance of the seeds designed as explained in Chapter 9, these algorithms are implemented, along with the seeding concepts presented previously, in an experimental read mapping software tool which is used for the tests presented later in Chapter 11.

10.1 Setup

When mapping a read to a reference genome, one expects little difference between the sequenced data and the reference data. For this reason, it makes sense to avoid building a full dynamic programming alignment table and explore all possible alignments, and compute only a bandwidth alignment table instead. The advantages are twofold: first, bandwidth alignment is significantly less time-consuming than full alignment, and second, alignments with many insertions / deletions are discarded from the start, and the algorithm obtains significant alignments revealing important conservation between the reads and the reference, if such alignments exist. The actual "width" of the bandwidth alignment is from a parameter k which gives the maximum number of insertions or deletions accepted in an alignment. As such, the number of diagonals in the dynamic programming table that are actually computed by the alignment algorithm is $2 \cdot k + 1$.

Let m be the read length, and let i and j be the positions of a hit by a seed π on the read $R = r_1...m$ and the reference sequence $S = s_1...s_N$ respectively. $S, R \in \{\mathsf{B},\mathsf{G},\mathsf{Y},\mathsf{R}\}^*$, with colors encoded internally as the integers $0, 1, 2, 3$ as explained earlier in Section 8.1. Note that, like in the previous chapter, the *color translation of the reference sequence* is used by the alignment algorithms. The single-hit strategy is considered, i.e. one seed hit is sufficient to trigger the further processing. Then, any alignment with at most k indels that contains the matches detected by the seed will involve a subsequence of the reference that is contained

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

between positions $j - i - k$ and $j - i + m + k$. Let T be the subsequence $S_{(j-i-k)..(j-i+m+k)}$ of the reference, of length $m + 2k$. To extend the hit, the bandwidth alignment is performed in color space between the read R and the subsequence T of the reference.

10.2 SIMD filtration

A SIMD bandwidth alignment algorithm acts as a second filter, with the goal of eliminating most of the false positive hits of the spaced seeds. It can process several hits in a single run, and performs a classic bandwidth sequence alignment, with no built-in mechanism for handling the properties of the color encoding. The pairs of color sequences that align with a score above an established threshold θ (by default, which have at least 70% color matches) are further submitted to the second alignment algorithm, while the sequences that do not share sufficient color similarity are discarded.

Technically, this filter uses the `sse2` instruction set and implements the local alignment algorithm of Gotoh [75] on *compressed* data (2 bits per color). For example, on reads of length 64 aligned with at most 7 indels (16 diagonals), the filter processes 2 millions hits in less than 1 second, using only one core of a 2.57 GHz Core2 Dual processor. We noticed that the optimal speed is reached when processing pairs of hits in a single run (the four hits per runs and the single hit per run being a little bit slower). In our case, 1 million pairs of hits (of reads of length 64) are processed in less than 1 second on a 2.57 Ghz Core 2 dual; at 2048 cells per pair of hits, this gives 2 billions SW-cells per second, which is comparable with the full SW version implemented in [60] and [179] (3 billions SW-cells and 0.5 billions SW-cells). Note that the "practical" speedup heavily depends on the fact that only the bandwidth is computed in our implementation.

10.3 "Base-intelligent" algorithm

Biological variations in color space (revisited) The goal of this algorithm is to obtain alignments of color sequences that are meaningful at the nucleotide level, i.e. to make the distinction between mismatches caused by reading errors and those caused by SNPs and indels, and to properly assign the corresponding scores and penalties. This can be done thanks to the interpretation of colors as *transformations* of bases [1] (for example, the color R transforms C in G, as shown earlier by Figure 8.1, Section 8.1). From this perspective, valid DNA modifications correspond to an alignment of different color sequences that "transform" a base symbol A into another base symbol B, passing through different intermediate symbols, as discussed earlier in Section 8.1 of Chapter 8 and illustrated by Figure 8.7. Consequently, it does not suffice to align pairs of colors to detect the changes at the DNA level. Instead, colors need to be considered and evaluated in their context. To achieve this, the alignment of color sequences needs a *limited memory of the previous color mismatches* on each path of the alignment matrix, as will be explained below.

Handling reading errors on top of biological variations In addition to SNPs and indels defining DNA sequence differences between the read and the reference, sequencing errors affecting the read sequence must be handled by the alignment algorithm. Section 8.1 explained earlier how to distinguish between isolated reading errors and SNPs in a color sequence alignment. This task becomes more difficult if one wishes to handle more complex situations, such as sequences

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

of color matches and mismatches that could correspond, in theory, either to biological variations or to reading errors.

To answer this issue, the basic idea of the approach adopted by the alignment algorithm presented here is the following: *unless a color mismatch is "corrected" (followed by other mismatches that will eventually lead to the same base in both nucleotide sequences) within a number of steps, it is considered a reading error.* The algorithm is essentially a *heuristic* based on a *dynamic programming* approach, hybridized with a *greedy* evaluation of color mismatches as reading errors or SNPs based on paired colors within a certain distance. The main difference with respect to algorithms such as [49, 93, 179] discussed previously in Section 8.3.3 resides in the fact that this algorithm does not explore explicitly all possible translations of the read. The choice is basically motivated by the fact that the probability of having n consecutive SNPs in an alignment decreases drastically with the growth of n, hence the choice to label as "reading error" any mismatch not corrected within a small number of steps. Although this greedy approach does not guarantee the "correct" assessment of reading errors and SNPs, it handles situations that can occur in an alignment of sufficiently similar color sequences.

Let M be the dynamic programming table storing the scores associated to partial color sequence alignments between the read R and the reference fragment T, with $M[i,j]$ giving the score of the best semi-global alignment between the prefix $R_{1..i}$ of the read and the prefix $T_{1..j}$ of the reference fragment. Three additional bandwidth tables are maintained in order to ensure the availability of context information for each of the explored alignment paths:

- C holds, for each partial alignment, a *color code* which basically states whether the current alignment is "in phase", meaning that all previous color mismatches have been either canceled by subsequent mismatches as being part of SNPs, or identified as reading errors and corrected manually. As such, $C[i,j] =$B when all mismatches on this alignment path have been identified as either SNPs or reading errors, and $C[i,j] \neq$B when the sources of the latest mismatch(es) have not yet been assessed. In principle, in the absence of identified reading errors, $C[i,j] = r_1 \oplus ... \oplus r_i \oplus t_1 \oplus ... \oplus t_j$, where r_i is the color at the ith position of the read R, and t_i is the color at the ith position of the reference fragment T. Whenever a misread color r_h is detected a position h of the read, then the correct value of r_h is assumed to be the color $t_{h'}$ to which it is aligned.

- H holds a *limited history* of the mismatches having occurred on each aliment path. Let (r_i, t_j) be a pair of aligned symbols, with $(r_i, t_j) \in \{$B,G,Y,R,$-\}^2 \setminus \{(-,-)\}$. Then their alignment can be captured by a color h as follows:

$$h_{i,j} = \begin{cases} r_i \oplus t_j & \text{if } r_i \neq - \text{ and } t_j \neq - \\ r_i & \text{if } t_j = - \\ t_j & \text{if } r_i = -. \end{cases} \tag{10.1}$$

  If $H[i,j] = h_{i,j}$ defined as above is set to a value other than B, this value is propagated in subsequent cells belonging to alignments that pass through $[i,j]$ (i.e., if $H[i-1, j-1] \neq$B and the alignment ending at $[i-1, j-1]$ passes through $[i,j]$, then $H[i,j]$ becomes $H[i-1, j-1]$ instead of being computed according to relation (10.1). The history is reset to B once the nature of the (r_i, t_j) mismatch is determined in a larger context, as will be explained below.

The algorithm is given by relations (10.2) - (10.5). After a classic initialization for semi-global

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

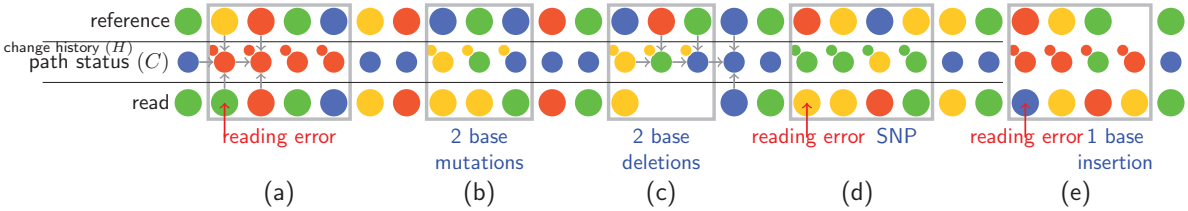

Figure 10.1: Examples of SNP, reading error and indel detection on an alignment path. **(a)** An isolated reading error corresponds to a color mismatch that remains uncorrected for several steps, after which the values of C and H are reset to B. **(b)** Two consecutive nucleotide mutations correspond to two sequences of three colors. The last of the three color mismatches changes C to B and is therefore not penalized (according to (10.4 b). The history is reset since the sources of mismatches have been identified. **(c)** Two consecutive nucleotide deletions correspond to two color deletions preceded (in this case) by a color change. The last nucleotide paired with a gap changes C to B (case handled by (10.5 a). The history is reset since the sources of mismatches have been identified. **(d)** In case of a reading error followed by a SNP, the last color mismatch of the SNP changes C to match H. The sequence is recognized as a SNP and therefore the last mismatch is not penalized (according to (10.4 b), in order to account for a single mismatch at the DNA level. The color change introduced by the preceding reading error is eliminated from C and H is reset. **(e)** Similarly to (d), in case of a reading error followed by a nucleotide deletion, the last nucleotide paired with a gap changes C to match H (case handled by (10.5 a). The C and H are reset since the sources of mismatches have been identified.

alignment, the score for each partial solution is computed as shown by relation (10.2).

$$M[i,j] = \max \begin{cases} M[i-1,j-1] + score(r_i, t_j, C[i-1,j-1], H[i-1,j-1]); \\ M[i,j-1] + gapPenalty(t_j, C[i,j-1], H[i,j-1]); \\ M[i-1,j] + gapPenalty(r_i, C[i-1,j], H[i-1,j]). \end{cases} \tag{10.2}$$

The score for aligning a pair of colors depends not only on the aligned colors, but also on the mismatches having occurred previously, which are captured by H and C, with H handled as explained above and C basically given by relation (10.3), where $[i_{prev}, j_{prev}]$ are the coordinates of the entry chosen in (10.2) for obtaining $M[i,j]$.

$$C[i,j] = \begin{cases} C[i-1,j-1] \oplus r_i \oplus t_j & \text{if } [i_{prev}, j_{prev}] = [i-1, j-1]; \\ C[i,j-1] \oplus t_j & \text{if } [i_{prev}, j_{prev}] = [i, j-1]; \\ C[i-1,j] \oplus r_i & \text{if } [i_{prev}, j_{prev}] = [i-1, j]. \end{cases} \tag{10.3}$$

The scoring function is given by relations (10.4) and(10.5). Options (a), (c) of (10.4) and (b) of (10.5) basically correspond to classic match, mismatch and gap scores respectively. In addition to these, some special cases are considered. This scoring function *does not penalize* mismatches of colors that "correct" the color on a path, such as the last mismatch in a series corresponding to SNPs – (10.4 b), or the mismatch that is part of a base indel – option (10.5 a). This choice of scoring function is based on the observation (illustrated earlier in Figure 8.7 a and c) that n consecutive nucleotide modifications can affect $n+1$ color pairs. As such, one of the $n+1$ penalties must be eliminated. More precisely, option (10.4 b) assigns score 0 to the last mismatch of a sequence corresponding to a series of SNPs, while option (10.5 a) adds the absolute value of a mismatch to the gap penalty in order to compensate for the mismatch (if any) preceding the color indel. Such a color correction is detected by the fact that it either resets C to the neutral color B, or it changes C to match the first color change in the series, stored in H. The latter situation basically identifies this initial color change as a reading error, followed by one

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

or more color mismatches corresponding to SNPs or being part of indels, As such, this kind of color correction *triggers the reset of both* $C[i,j]$ *and* $H[i,j]$ *to* B. $C[i,j]$ and $H[i,j]$ are also reset if the color mismatch indicated by $H[i,j]$ is not corrected within a number of steps.

$$score(r,t,c,h) = \begin{cases} match & \text{if } r = t; & (a) \\ 0 & \text{if } r \neq t \text{ and } (c \oplus r \oplus t = \mathsf{B} \text{ or } c \oplus r \oplus t = h); & (b) \\ -mismatch & \text{otherwise.} & (e) \end{cases} \tag{10.4}$$

$$gapPenalty(s,c,h) = \begin{cases} -d + mismatch & \text{if } (c \neq \mathsf{B} \text{ and } c \oplus s = \mathsf{B}) \text{ or } (c \neq h \text{ and } c \oplus s = h); & (a) \\ -d & \text{otherwise.} & (b) \end{cases} \tag{10.5}$$

When read qualities are available, match and mismatch scores depend on the quality of the paired read position, in that their absolute value is proportionally decreased for poor quality colors, which hold less reliable information.

Various color mismatch situations and the way they are handled within the presented algorithm are illustrated by Figure 10.1.

Perspectives Primarily developed only as an instrument for evaluating the significance of the designed seeds, this algorithm can further be improved via a more rigorous scoring scheme and mismatch source assessment (i.e. whether color mismatches correspond to reading errors or DNA variations) by managing a more advanced mismatch history.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Chapter 11

Experiments and discussion

This chapter presents and discusses several experiments meant to evaluate the efficiency of the seeds designed using the method of Chapter 9. These were performed using our experimental read mapping tool SToRM, which uses positioned seeds and implements the hit extension algorithms mentioned in the previous chapter.

11.1 Comparison with other methods

The performance of the proposed seeds is compared here with that of MAQ 0.7.1 [123], SHRiMP 1.3.2 [179] and 2.0, and PerM 0.2.6 [37] which are popular software tools for mapping SOLiD reads using the seed approach, and with two tools implementing the Burrows-Wheeler Transform approach: Bowtie 0.12.5 [115], and BWA 0.5.7 [122]. The tools were tested with their default settings, on a machine with 8 Intel Xeon CPUs running at 2GHz and 4G RAM.

Since the emphasis of this experiment is on seed accuracy, we run our implementation (SToRM) with the spaced seeds proposed by other seed-based tools in addition to the seeds designed with our method, in order to establish the quality of all the seed families in the same setup.

Table 11.1 shows seed families used in our experiments. SHRiMP-DEFAULT is the default set of SHRiMP. PERM-F3-S20 is composed of seeds F_3 and $S_{2,0}$ of [37] taken together. Seeds 3-LOSSY-12 and 3-LOSSLESS-10-24P are two seed families of weights 12 and 10 respectively, designed with our method, using the following parameters for the underlying model: SNP probability 0.0085, indel probability 0.0015, reading error probability at the beginning of the read 0.01, reading error probability at the end of the read 0.1, periodic reading error probability 0.02. 3-LOSSY-12 is the default seed family for SToRM, and 3-LOSSLESS-10-24P are positioned seeds (24 positions) lossless for 1 indel and 1 SNP, 1 indel and 1 reading error, 2 SNPs, or 1 SNP and 2 reading errors.

11.1.1 Read-mapping tools comparison

Tables 11.2 and 11.3 show the results of experiments run respectively on 1,280,536 reads from *Saccharomyces cerevisiae* mapped on a 12,160,680 bp genome, and on 1,000,000 and reads from *Escherichia coli* mapped on a 4,573,347 bp genome (the latter is a public dataset available on the Applied Biosystems website).

We can see that the approaches based on the Burrows-Wheeler Transform, Bowtie and BWA, are very fast but less accurate than most of the other tools. PerM is by far the fastest among

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Seed set ID	Patterns	Positions
SHRiMP-DEFAULT	11111***1111111	All
	1111**111**1**1111	
	111**1**1***111**1111	
	1111**1****1***1**1**1111	
PERM-F3-S20	111*1**1***111*1**1***11	All
	1111**1****1111**1****11	
3-LOSSY-12	1111*1111*1111	All
	1111*111**1****1111	
	1111****11**11*1111	
3-LOSSLESS-10-24P	1111*11*1111	0,1,2,3,4,5,6,7,8,18,19,20
	1*11111111*1	2,12,15,16,18,19,20,21
	1111*1*******1*1111	0,1,11,14

Table 11.1: Seed families used in the experiments.

Program	Mapped reads	Unique mapping positions	Execution time
BOWTIE	553,140 (43.20%)	512,086 (39.99%)	0m50s
BWA	422,550 (33.00%)	395,342 (30.87%)	0m38s
MAQ	616,497 (48.14%)	567,549 (44.32%)	1m20s
PERM	418,524 (32.68%)	347,668 (27.15%)	0m31s
SHRiMP 1.3.2	663,923 (51.85%)	-	7m56s
SHRiMP 2.0	709,146 (55.38%)	-	1m22s
STORM	839,633 (65.57%)	754,402 (58.91%)	2m10s

Table 11.2: Comparison of 6 read-mapping tools on the *S. cerevisiae* dataset. The execution time refers to all steps, including index construction for the tools that can reuse the result of this step (Bowtie, BWA, MAQ).

Program	Mapped reads	Unique mapping positions	Execution time
BOWTIE	456,416 (45.64%)	423,541 (42.35%)	0m18s
BWA	456,928 (45.69%)	424,239 (42.42%)	0m21s
MAQ	646,523 (64.65%)	588,362 (58.84%)	1m08s
PERM	413,102 (41.31%)	384,050 (38.40%)	0m23s
SHRiMP 1.3.2	687,855 (68.79%)	-	3m33s
SHRiMP 2.0	714,662 (71.47%)	-	0m41s
STORM	773,155 (77.32%)	697,164 (69.72%)	0m57s

Table 11.3: Comparison of 6 read-mapping tools on the *E. coli* dataset. The execution time refers to all steps, including index construction for the tools that can reuse the result of this step (Bowtie, BWA, MAQ).

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Seed family	Mapped reads	Unique mapping positions	Execution time
PerM-F3-S20	768,732 (60.03%)	694,951 (54.27%)	0m55s
SHRiMP-default	836,899 (65.36%)	751,761 (58.71%)	2m15s
3-lossy-12	839,633 (65.57%)	754,402 (58.91%)	2m10s
3-lossless-10-24p	839,072 (65.53%)	755,208 (58.98%)	2m06s

Table 11.4: Comparison of 4 different seed families on the *S. cerevisiae* dataset.

Seed family	Mapped reads	Unique mapping positions	Execution time
PerM-F3-S20	731,447 (73.14%)	662,666 (66.27%)	0m33s
SHRiMP-default	772,861 (77.29%)	696,903 (69.69%)	1m02s
3-lossy-12	773,155 (77.32%)	697,164 (69.72%)	0m57s
3-lossless-10-24p	772,336 (77.23%)	696,615 (69.66%)	0m44s

Table 11.5: Comparison of 4 different seed families on the *E. coli* dataset.

the seed-based tools, thanks to its efficient periodic pattern indexing, but on the downside, it has a very poor sensitivity. This is caused by the fact that PerM seeds are designed lossless with respect to mismatches and SNPs but do not deal well with indels, most likely because of their large span. Note also that the quality of lossless seeds is mediocre on alignments exceeding their error threshold, as illustrated previously by Figure 9.12, Section 9.4. Finally, SToRM has an advantageous percentage of mapped reads and execution time tradeoff among the seed-based tools.

11.1.2 Seed families comparison

We further focused on the performance of the seed families described in Table 11.1, which we tested within our implementation on the aforementioned data sets. The test setup was identical for all seed families and consisted of the default SToRM parameters: the scores for match, mismatch, gap opening and gap extension were 5, -4, -6, -4 respectively, seeds were used with a single-hit strategy, the acceptance threshold score for the SIMD filter was 100, and obtained alignments were considered significant above the score 115. The results are given in Tables 11.4 and 11.5.

The PerM seeds have the smallest sensitivity, which is due, as mentioned above, to their large span and the fact that they are lossless for a small number of errors. These seeds would perform very well on data with few reading errors and no indels, but are unsuited for data with higher variations.

Comparing, within our program, the default seeds of SHRiMP of weight 12 (SHRiMP-default) with those with the same weight designed using our method, we observe a higher sensitivity of our seeds, reached with a smaller number of seeds, hence with less memory used for the index.

Finally, the family of positioned seeds of weight 10 (3-lossless-10-24p) has a performance comparable to that of 3-lossy-12 in number of mapped reads, thus illustrating the advantageous sensitivity/selectivity tradeoff of positioned seeds depicted earlier in Figure 9.11, and commented in Section 9.4. More precisely, we notice that the positioned seeds performed slightly better than 3-lossy-12 on the *S. cerevisiae* dataset, and slightly worse on the *E. coli* dataset, most

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

likely because the periodic reading error bias is less obvious on the latter. Hence, we are confident that position-restricted seeds can be beneficial on data with a higher variation of reading error level, as well as for data sets with larger read lengths. Furthermore, while the execution time improvements brought by 3-LOSSLESS-10-24P in comparison with 3-LOSSLESS-12 are not impressive, the benefits are significant on the index size, since all the keys for a seed of weight 10 can be stored in 16 times less memory that the keys for a seed of weight 12.

11.2 Conclusions and perspectives

The main focus of this work is a seed design framework for mapping SOLiD reads to a reference genomic sequence. The contributions include the concept of position-restricted seeds, particularly suitable for short alignments with non-uniform error distribution; a model that captures the statistical characteristics of the SOLiD reads, used for the evaluation of lossy seeds; an efficient dynamic programming algorithm for verifying the lossless property of seeds; the ability to distinguish between SNPs, reading errors and indels in seed design; an algorithm for aligning color sequences with the possibility of handling overlapping mutations and reading errors; an experimental read mapping tool that implements these concepts.

Further work will include a more rigorous training of our models and in particular a more accurate estimation of involved probabilities, possibly using advanced methods of assessing the fit of a model. Additionally, the IEDERA framework can benefit from an extension allowing the design of seed famlies for multiple hit stategies, i.e. the evaluation of the sensitivity when several hits are required before proceding to the alignment. Another interesting question to study is the design of efficient combined lossy/lossless seeds which provide a guarantee to hit all the alignments with a specified number of errors and still have a good sensitivity when this threshold is exceeded. However, since lossless seeds tend to have a regular structure (see [109]) while best lossy seeds often have asymmetric and irregular structure, computing such seeds may be difficult.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Published papers and talks

Journal paper (2010):

Noé, L., Gîrdea, M., and Kucherov, G. (2010). Designing spaced seeds for SOLiD read mapping. *Advances in Bioinformatics, Volume 2010.*

Conference paper (2010):

Noé, L., Gîrdea, M., and Kucherov, G. (2010). Seed design framework for mapping SOLiD reads. In *Proceedings of the 14th Annual International Conference on Computational Molecular Biology (RECOMB).*

Talk (2010):

Gîrdea, M. (2010). Seed design framework for mapping SOLiD reads. *GTSeq 2010*, January 25-26, Montpellier, France.

Poster (2009):

Gîrdea, M., Noé, L., and Kucherov, G. (2009). Read mapping tool for AB SOLiD data. *9th International Workshop in Algorithms in Bioinformatics (WABI), Philadelphia (USA).*

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Concluding remarks

This thesis focused on the design of exact and heuristic alignment algorithms, and scoring schemes answering two complex sequence similarity problems: i) the detection of hidden protein homologies *by protein sequence comparison*, when the source of the divergence are frameshift mutations, and ii) *mapping short SOLiD reads* (sequences of overlapping di-nucleotides encoded as colors) to a reference genome.

The first problem was addressed in **Part II**, from a codon evolution perspective. Common origins of proteins are sought by implicitly aligning all their putative coding DNA sequences, stored in efficient data structures called back-translation graphs. The triplet nature of coding DNA is captured in the dynamic programming algorithm designed for aligning these graphs and in its scoring scheme. Unlike classic sequence alignment algorithms, this approach does not make the assumption of independence between pairs of aligned symbols. Instead, the translation-dependent scoring function for nucleotide pairs, ensures that the obtained alignments reflect the expected dynamics of coding DNA sequences.

For the second problem, **Part III** proposes algorithmic solutions which incorporate knowledge about the characteristics of the SOLiD encoding and the observed artifacts of this sequencing technology, both captured by probabilistic models of read alignments. The dually encoded nucleotide information in adjacent colors is handled by implicit interpretation of color sequences as DNA, also taking into account possible reading errors. In this part, the focus is mainly on designing efficient spaced seeds for mapping such color-encoded reads. These seeds allow identifying mapping positions for the reads on the reference genome, which is is a crucial step with effects on both the quality and the speed of the read mapping process.

In conclusion, in both cases, the same general idea was applied: to implicitly compare DNA sequences for detecting changes occurring at this level, while manipulating, in practice, other representations (protein sequences represented as graphs, sequences of di-nucleotide codes) that provide additional information and thus help to improve the similarity search. The thesis proposes generic methods for approaching the respective problems, as well as implementations of these methods, freely available for download or online use.

This work opens new perspectives in designing models and algorithms adapted to specific sequence alignment problems, and supports the idea of understanding and making full use of the statistical characteristics of the aligned sequences, by capturing them in specially designed alignment algorithms, scoring functions, and adapted seeding concepts.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Glossary

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

α **helix** A common motif in the secondary structure of proteins. A coiled or spiral conformation, in which every backbone NH group created a hydrogen bond with the backbone CO group of the amino acid four positions earlier. 22

β **sheet** A form of regular secondary structure in proteins consisting of parallel or antiparallel strands in which the NH groups in the backbone of one strand establish hydrogen bonds with the CO groups in the backbone of the adjacent strand. 22

adapter A short, chemically synthesized DNA molecule which is used to link the ends of two other molecules. 33, 34

allele A variant of the DNA sequence at a given locus. 19, 30

amino acid Molecules containing an amine group, a carboxylic acid group and a side chain. Amino acids are the structural units of proteins. 22, 24–26, 29, 39–42, 47–51, 53, 63, 69–72, 75–79, 83, 90, 91, 93, 98–100, 102–105, 108

amplicon A short DNA sequence whose multiple copies are a product of the amplification process of the polymerase chain reaction. 33, 34

anticodon A sequence of 3 nucleotides in the tRNA that can form complementary base pairs with one or more codons in the mRNA which encode a specific amino acid. 26

antiparallel Two molecules are antiparallel if they run side-by-side in opposite directions. 20, 22, 28

backbone In a polymer, the covalently bonded atoms that create the continuous chain of the molecule. 20, 22

biopolymer Polymer produced by a living organism. 24

cell The functional basic unit of all known living organisms. It is the smallest unit of life that is classified as a living thing. 19, 25

chromosome An organized structure of DNA and protein. A chromosome consists of a single piece of coiled DNA composed of regions with different purposes such as genes or regulatory elements, along with DNA-bound proteins, which serve to package the DNA and control its functions. 19

codon Tri-nucleotide sequence in coding DNA, which encodes an amino acid or marks the termination of the coding sequence. 25, 26, 29, 40, 49–51, 67, 69–72, 75, 76, 90, 91, 93, 98, 100

cytoplasm The part of a cell that is enclosed within the cell membrane. 19, 25

deoxynucleotide A nucleotide containing a deoxyribose sugar. Deoxyibonucleotides are found in DNA molecules. 21, 32

dideoxynucleotide A nucleotide containing a dideoxyribose sugar. Dideoxyribonucleotides are used as chain terminators in Sanger sequencing methods. 32

DNA deoxyribonucleic acid. 19–21, 24–26, 28, 31, 39, 49, 53, 67, 68, 72, 75, 119, 128, 142

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

enzyme Protein that catalyzes (i.e., increase the rates of) chemical reactions. 21, 24, 28, 36

eukaryote Organism whose cells contain complex structures inside the membranes. The defining membrane-bound structure that sets eukaryotic cells apart from prokaryotic cells is the nucleus, or nuclear envelope, within which the genetic material is carried. Most eukaryotic cells also contain other membrane-bound organelles such as mitochondria, chloroplasts and the Golgi apparatus. Almost all species of large organisms are eukaryotes, including animals, plants and fungi. 19, 22, 25, 26

exon A region of a transcribed gene present in the final functional RNA molecule. 25

folding The process by which a molecule assumes its shape or conformation. 24

frameshift Insertion or deletion of a number of nucleotides that is not evenly divisible by 3 in a coding sequence, which disrupts gene expression by codons. 14, 15, 29, 30, 67–73, 75, 80, 81, 83, 86–88, 90, 91, 93, 97, 99, 102–104, 106–109, 114

gene A segment of DNA corresponding to a "unit of inheritance" that codes for a type of protein or for an RNA chain that has a function in the organism. 19, 24, 25, 30, 67

genome The complete genetic information of an organism. 19

hydrogen bond The attractive interaction of a hydrogen atom with an electronegative atom, like nitrogen, oxygen or fluorin. The hydrogen must be covalently bonded to another electronegative atom to create the bond. These bonds can occur between molecules (intermolecularly), or within different parts of a single molecule (intramolecularly). The hydrogen bond is weaker than covalent or ionic bonds. 20, 22, 28

indel Insertion or deletion of one or more structural units from a polymer (nucleotides from a nucleic acid sequence, or amino acids from a protein sequence). 29, 40, 41, 63, 72, 119, 123, 125, 127–129, 131, 142

intron A DNA region within a gene that is not translated into protein. 25

ligase Enzyme that catalyses the joining of two large molecules by forming a new chemical bond. 28, 33

locus Specific location of a gene or DNA sequence on a chromosome. 19, 30

macromolecule A very large molecule most often created by some form of polymerization. 19, 22

mitochondrion A membrane-enclosed organelle found in most eukaryotic cells. Mithocondria supply cellular energy, and are involved in several processes such as signaling, cellular differentiation, cell death, the control of the cell cycle and cell growth. The mitochondrion has its own independent genome. 19

mRNA messenger RNA. 21, 25, 26

mutation Modification in the DNA sequence of a cell's genome. 19, 29, 39–42, 47, 49, 50, 67–69, 71

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

nucleoside A molecule composed of a nucleobase (nitrogenous base) and a five-carbon sugar (either ribose or 2'-deoxyribose). 20

nucleotide A molecule composed of a nucleobase (nitrogenous base), a five-carbon sugar (either ribose or 2'-deoxyribose), and one to three phosphate groups. Nucleotides are the structural units of nucleic acids (DNA and RNA). 19–21, 25, 28, 29, 31–34, 36, 39–42, 49, 69, 72, 75, 78, 80, 98, 127

nucleus A membrane-enclosed organelle found in eukaryotic cells. It contains most of the cell's genetic material, organized as multiple long linear DNA molecules in complex with a large variety of proteins, such as histones, to form chromosomes. 19, 25

oligonucleotide A short sequence of nucleotides, typically with twenty or fewer bases. 34

organelle A specialized subunit within a cell that has a specific function, and is usually separately enclosed within its own membrane. 19

PCR polymerase chain reaction. 28, 32–34

peptide bond Chemical bond formed between two molecules when the carboxyl group of one molecule reacts with the amine group of the other molecule, releasing a molecule of water. 22

polymer A large molecule (macromolecule) composed of repeating structural units. 20, 21

polymerase Enzyme whose primary function is the polymerization of new DNA or RNA against an existing DNA or RNA template in the processes of replication and transcription. 24, 28, 33, 36

primer A strand of nucleic acid (a sequence of nucleotides) that serves as a starting point for DNA or RNA synthesis. Primers are required because the enzymes that catalyze replication, DNA polymerases, can only add new nucleotides to an existing strand of DNA ant are not capable of initiating a strand. 28, 31–34

prokaryote Organism that lacks a cell nucleus (= karyon), or any other membrane-bound organelles. Most prokaryotes are unicellular, with a few exceptions (e.g. myxobacteria) which have multicellular stages in their life cycles. Domains: the bacteria and the archaea. 19, 25

protein Organic compound made of amino acids arranged in a linear chain and folded into a globular form. 19, 21, 22, 24–26, 28, 36, 39, 40, 53, 67, 71, 72, 75

purine A basic compound, composed of two fused heterocyclic rings. The purines occuring in nucleic acids are *adenine* (A) and *guanine* (G). 20, 29, 40, 41, 70, 120

pyrimidine A nitrogen-containing, single-ring, basic compound. The pyrimidines that occurs in DNA are *cytosine* (C) and *thymine* (T). In RNA, the thymine is replaced by *uracil* (U). 20, 29, 40, 41, 70, 106, 120

read mapping The process of finding matching segments on a reference genome for a set of short reads obtained from DNA sequencing. 37, 128

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

reading frame The contiguous and non-overlapping set of 3 nucleotide codons in a coding DNA sequence. 25, 29, 67, 69, 71, 90, 92, 97, 100

ribonucleotide A nucleotide containing a ribose sugar. Ribonucleotides are found in RNA molecules. 24

ribosome The components of cells that build proteins from amino acids, by binding to a messenger RNA (mRNA) and using it as a template for the sequence of amino acids in a particular protein. The amino acids are attached to transfer RNA (tRNA) molecules, which enter one part of the ribosome and bind to the mRNA sequence. The attached amino acids are then joined together, and the ribosome moves along the mRNA, "reading" its sequence and producing the amino acid chain. 26

RNA ribonucleic acid. 19, 21, 24–26, 39

SNP single nucleotide polymorphism. 29, 119, 123, 125, 126, 128, 129, 142

splicing A modification of an RNA after transcription, in which introns are removed and exons are joined. 25

transcription The process of creating an equivalent RNA copy of a DNA sequence. 19, 24, 25, 29

transition A point mutation that changes a purine nucleotide to another purine (A $\leftrightarrow$ G) or a pyrimidine nucleotide to another pyrimidine (C $\leftrightarrow$ T). 29, 41, 49, 50, 63, 71

translation The process of interpreting the information encoded in a messenger in order to produce a specific protein. 19, 25, 29, 72

transversion The substitution of a purine for a pyrimidine or a pyrimidine for a purine. 29, 41, 49, 50, 63, 71

tRNA transfer RNA. 21, 26

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

Bibliography

[1] ABI. A theoretical understanding of 2 base color codes and its application to annotation, error detection, and error correction. methods for annotating 2 base color encoded reads in the SOLiD^{TM}system, 2008.

[2] C. Adessi, G. Matton, G. Ayala, G. Turcatti, J.J. Mermod, P. Mayer, and E. Kawashima. Solid phase DNA amplification: characterisation of primer attachment and amplification mechanisms. *Nucleic Acids Research*, 28(20):e87, 2000.

[3] A.V. Aho and M.J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):340, 1975.

[4] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*, chapter 7: How Genomes Evolve. Garland Science, 2002.

[5] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*, chapter 4: Chromosomal DNA and Its Packaging in the Chromatin Fiber. Garland Science, 2002.

[6] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*, chapter 6: How Cells Read the Genome: From DNA to Proteins. Garland Science, 2002.

[7] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*, chapter 5: DNA Replication Mechanisms. Garland Science, 2002.

[8] C.L. Alston, M. Morak, C. Reid, I.P. Hargreaves, S.A.S. Pope, J.M. Land, S.J. Heales, R. Horvath, H. Mundy, and R.W. Taylor. A novel mitochondrial MTND5 frameshift mutation causing isolated complex I deficiency, renal failure and myopathy. *Neuromuscular Disorders*, 2009.

[9] S.F. Altschul. Evaluating the statistical significance of multiple distinct local alignments. *Theoretical and Computational Methods in Genome Research*, pages 1–14, 1997.

[10] S.F. Altschul, R. Bundschuh, R. Olsen, and T. Hwa. The estimation of statistical parameters for local alignment score distributions. *Nucleic Acids Research*, 29(2):351–361, 2001.

[11] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *JMB*, 215(3):403–410, 1990.

[12] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[13] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, 1997.

[14] M.A. Armstrong. *Groups and symmetry*, chapter 10: Products. Springer, 1988.

[15] A.N. Arslan. Regular expression constrained sequence alignment. *Journal of Discrete Algorithms*, 5(4):647–661, 2007.

[16] L. Arvestad. Aligning coding DNA in the presence of frame-shift errors. *Proceedings of the 8th Annual CPM Symposium*, 1264:180–190, 1997.

[17] L. Arvestad. *Algorithms for biological sequence alignment*. PhD thesis, Royal Institute of Technology, Stocholm, Numerical Analysis and Computer Science, 2000.

[18] D.R. Bentley, S. Balasubramanian, H.P. Swerdlow, G.P. Smith, J. Milton, C.G. Brown, K.P. Hall, D.J. Evers, C.L. Barnes, H.R. Bignell, et al. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456(7218):53–59, 2008.

[19] J.M. Berg, J.L. Tymoczko, and L. Stryer. *Biochemistry*, chapter 5: DNA, RNA, and the Flow of Genetic Information. Freeman and Company: New York, 2002.

[20] J.M. Berg, J.L. Tymoczko, and L. Stryer. *Biochemistry*, chapter 3: Protein Structure and Function. Freeman and Company: New York, 2002.

[21] J.M. Berg, J.L. Tymoczko, and L. Stryer. *Biochemistry*, chapter 27: DNA Replication, Recombination, and Repair. Freeman and Company: New York, 2002.

[22] O.R.P. Bininda-Emonds. transAlign: using amino acids to facilitate the multiple alignment of protein-coding DNA sequences. *BMC Bioinformatics*, 6:156, 2005.

[23] R.D. Blake, S.T. Hess, and J. Nicholson-Tuell. The influence of nearest neighbors on the rate and pattern of spontaneous point mutations. *JME*, 34(3):189–200, 1992.

[24] J.L. Bobadilla, M. Macek, J.P. Fine, and P.M. Farrell. Cystic fibrosis: A worldwide analysis of CFTR mutations – correlation with incidence data and application to screening. *Human mutation*, 19(6):575–606, 2002.

[25] D. Branton, D.W. Deamer, A. Marziali, H. Bayley, S.A. Benner, T. Butler, M. Di Ventra, S. Garaj, A. Hibbs, X. Huang, et al. The potential and challenges of nanopore sequencing. *Nature biotechnology*, 26(10):1146–1153, 2008.

[26] B. Brejová, D.G. Brown, and T. Vinar. Optimal spaced seeds for Hidden Markov Models, with application to homologous coding regions. In *Proceedings of the 14th Symposium on Combinatorial Pattern Matching (CPM)*, volume 2676 of *LNCS*, pages 42–54. Springer, 2003.

[27] B. Brejová, D.G. Brown, and T. Vinar. Vector seeds: An extension to spaced seeds. *Journal of Computer and System Sciences*, 70(3):364–380, 2005.

[28] D.G. Brown. A survey of seeding for sequence alignment. *Bioinformatics algorithms: techniques and applications*, pages 117–142, 2008.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[29] J. Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.

[30] J. Buhler, U. Keich, and Y. Sun. Designing seeds for similarity search in genomic DNA. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology (RECOMB), Berlin (Germany)*, pages 67–75. ACM Press, April 2003.

[31] S. Burkhardt and J. Kärkkäinen. Better filtering with gapped q-grams. *Fundamenta Informaticae*, 56(1-2):51–70, 2003. Preliminary version in Combinatorial Pattern Matching 2001.

[32] S. Burkhardt and J. Kärkkäinen. Better filtering with gapped q-grams. *Fundamenta Informaticae*, 56(1,2):51–70, 2003. Preliminary version in CPM 2001.

[33] A. Califano and I. Rigoutsos. FLASH: A fast look-up algorithm for string homology. In *1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93.*, pages 353–359, 1993.

[34] D. Campagna, A. Albiero, A. Bilardi, E. Caniato, C. Forcato, S. Manavski, N. Vitulo, and G. Valle. PASS: a program to align short sequences. *Bioinformatics*, 25(7):967–968, 2009.

[35] K.M. Chao and L. Zhang. *Sequence comparison: theory and methods*, chapter 8: Scoring matrices. Springer Verlag, 2008.

[36] K.M. Chao and L. Zhang. *Sequence comparison: theory and methods*, chapter 6: Anatomy of Spaced Seeds. Springer Verlag, 2008.

[37] Y. Chen, T. Souaiaia, and T. Chen. PerM: Efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics*, 25(19):2514–2521, 2009.

[38] F. Chiaromonte, V.B. Yap, and W. Miller. Scoring pairwise genomic sequence alignments. In *Pacific Symposium on Biocomputing 2002: Kauai, Hawaii, 3-7 January 2002*, page 115. World Scientific Pub Co Inc, 2001.

[39] Y.S. Chung, C.L. Lu, and C.Y. Tang. Efficient algorithms for regular expression constrained sequence alignment. *Information Processing Letters*, 103(6):240–246, 2007.

[40] J.M. Claverie. Detecting frame shifts by amino acid sequence comparison. *Journal of molecular biology*, 234(4):1140–1157, 1993.

[41] S.P. Cleary, H. Kim, M.E. Croitoru, M. Redston, J.A. Knight, S. Gallinger, and R. Gryfe. Missense polymorphisms in the adenomatous polyposis coli gene and colorectal cancer risk. *Diseases of the Colon & Rectum*, 51(10):1467–1474, 2008.

[42] A. Cornish-Bowden. IUPAC-IUB symbols for nucleotide nomenclature. *Nucleic Acids Res*, 13:3021–3030, 1985.

[43] G. Costa, C. Lee, L. Apone, J. Stuart, J. Warner, R. David, A. Sheridan, S. Ranade, J. Ichikawa, H. Peckham, S. McLaughlin, and K. McKernan. Applications of Next Generation Sequencing Using Stepwise Cycled-Ligation, 2007.

[44] F.H.C. Crick. The genetic code—yesterday, today, and tomorrow. In *Cold Spring Harbor Symp. Quant. Biol*, volume 31, pages 3–9, 1966.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[45] F.H.C. Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[46] F.H.C. Crick, L. Barnett, S. Brenner, and R.J. Watts-Tobin. General nature of the genetic code for proteins. *Nature*, 192:1227–1232, 1961.

[47] M. Csűrös and B. Ma. Rapid homology search with two-stage extension and daughter seeds. In *Computing and combinatorics: 11th annual international conference, COCOON 2005, Kunming, China, August 16-29, 2005: proceedings*, pages 104–114. Springer-Verlag New York Inc, 2005.

[48] M. Csűrös and B. Ma. Rapid homology search with neighbor seeds. *Algorithmica*, 48(2):187–202, 2007.

[49] M. Csurös, S. Juhos, and A. Bérces. Fast mapping and precise alignment of AB SOLiD color reads to reference DNA. In *Proceedings of the 10th international Workshop on Algorithms in Bioinformatics (WABI)*, LNCS, pages 176–188. Springer, 2010.

[50] G. Daniels. *Human blood groups*. Wiley-Blackwell, 2002.

[51] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5(Suppl 3):345–352, 1978.

[52] D.W. Deamer and M. Akeson. Nanopores and nucleic acids: prospects for ultrarapid sequencing. *Trends in biotechnology*, 18(4):147–151, 2000.

[53] D. Dressman, H. Yan, G. Traverso, K.W. Kinzler, and B. Vogelstein. Transforming single DNA molecules into fluorescent magnetic particles for detection and enumeration of genetic variations. *Proceedings of the National Academy of Sciences of the United States of America*, 100(15):8817, 2003.

[54] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, chapter 1. Cambridge Univ Pr, 1998.

[55] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, chapter 2: Pairwise alignment. Cambridge Univ Pr, 1998.

[56] M.S. Elliott and R.W. Trewyn. Inosine biosynthesis in transfer RNA by an enzymatic insertion of hypoxanthine. *Journal of Biological Chemistry*, 259(4):2407–2410, 1984.

[57] A. Elzanowski, J. Ostell, D. Leipe, and V. Soussov. The genetic codes. Bethesda, MD: National Center for Biotechnology Information (NCBI), 2000.

[58] B. Ewing and P. Green. Base-calling of automated sequencer traces using phred. II. error probabilities. *Genome Research*, 8(3):186–194, 1998.

[59] B. Ewing, L.D. Hillier, M.C. Wendl, and P. Green. Base-calling of automated sequencer traces using phred. I. accuracy assessment. *Genome Research*, 8(3):175–185, 1998.

[60] M. Farrar. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23(2):156–161, 2007.

http://doc.univ-lille1.fr

[61] M. Fedurco, A. Romieu, S. Williams, I. Lawrence, and G. Turcatti. BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies. *Nucleic Acids Research*, 34(3):e22, 2006.

[62] A. Fontaine and H. Touzet. Computational identification of protein-coding sequences by comparative analysis. In *Proceedings of the 1st IEEE international conference on Bioinformatics and Biomedecine (BIBM), Silicon Valley, California*, pages 95–102, 2007.

[63] R. Franklin and R.G. Gosling. Molecular configuration in sodium thymonucleate. *Nature*, 171(4356):740–741, 1953.

[64] B.G. Fry, H. Scheib, L. van der Weerd, B. Young, J. McNaughtan, S.F. Ryan Ramjan, N. Vidal, R.E. Poelmann, and J.A. Norman. Evolution of an arsenal: Structural and functional diversification of the venom system in the advanced snakes (caenophidia). *Molecular and Cellular Proteomics*, 7:215–246, 2008.

[65] A. Gambin, S. Lasota, R. Szklarczyk, J. Tiuryn, and J. Tyszkiewicz. Contextual alignment of biological sequences (Extended abstract). *Bioinformatics*, 18(Suppl 2):S116, 2002.

[66] C. Gehman, J. Labrenz, P. Baybayan, L.L. Cook, Q. Doan, A. Egan, G.A. Fry, R.A. Gibbs, C.L. Kovar, L.R. Lewis, E.R. Mardis, S.M. Moore, D.M. Muzny, A. Pradhan, S.J. Schneider, G.B.I. Scott, J. Sorenson, D.L. Steffen, D.M. Villasana, D. Wheeler, and S.P. Yang. Longer Reads with the KB Basecaller, 2004.

[67] E.M. Gertz, Y.K. Yu, R. Agarwala, A.A. Schäffer, and S.F. Altschul. Composition-based statistics and translated nucleotide searches: improving the TBLASTN module of BLAST. *BMC biology*, 4(1):41–54, 2006.

[68] M. Gîrdea, G. Kucherov, and L. Noé. Protein sequence alignment via anti-translation. *JOBIM 08*, pages 157–158, 2008.

[69] M. Gîrdea, G. Kucherov, and L. Noé. Back-translation for discovering distant protein homologies. In S.L. Salzberg and T. Warnow, editors, *Proceedings of the 9th International Workshop in Algorithms in Bioinformatics (WABI), Philadelphia (USA)*, volume 5724 of *Lecture Notes in Computer Science*, pages 108–120. Springer Verlag, September 2009.

[70] M. Gîrdea, G. Kucherov, and L. Noé. Back-translation for discovering distant protein homologies in the presence of frameshift mutations. *Algorithms for Molecular Biology*, 5(6), January 2010.

[71] M. Gîrdea, L. Noé, and G. Kucherov. Read mapping tool for AB SOLiD data. In *9th Workshop on Algorithms in Bioinformatics (WABI), September 12-13, 2009, Philadelphia (USA)*, September 2009. (poster).

[72] R. Giugno, A. Pulvirenti, M. Ragusa, L. Facciola, L. Patelmo, V. Di Pietro, C. Di Pietro, M. Purrello, and A. Ferro. Locally sensitive backtranslation based on multiple sequence alignment. In *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, (CIBCB)*, pages 231–237, 2004.

[73] N. Goldman and Z. Yang. A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Molecular Biology and Evolution*, 11(5):725–736, 1994.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[74] G.H. Gonnet. Back translation (protein to DNA) in an optimal way. Technical Report 505, Informatik, ETH, Zurich, December 2005.

[75] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705, 1982.

[76] R. Grantham, C. Gautier, M. Gouy, R. Mercier, and A. Pave. Codon catalog usage and the genome hypothesis. *Nucleic acids research*, 8(1):49–62, 1980.

[77] R.E. Green, A.W. Briggs, J. Krause, K. Prüfer, H.A. Burbano, M. Siebauer, M. Lachmann, and S. Pääbo. The Neandertal genome and ancient DNA authenticity. *The EMBO Journal*, 2009.

[78] R.E. Green, J. Krause, A.W. Briggs, T. Maricic, U. Stenzel, M. Kircher, N. Patterson, H. Li, W. Zhai, M.H.Y. Fritz, et al. A draft sequence of the Neandertal genome. *science*, 328(5979):710, 2010.

[79] X. Guan and E.C. Uberbacher. Alignments of DNA and protein sequences containing frameshift errors. *Bioinformatics*, 12(1):31–40, 1996.

[80] R. Guigó. DNA composition, codon usage and exon prediction. *Bishop M, editor. Genetic databases*, pages 53–80, 1999.

[81] E.J. Gumbel. *Statistics of extremes*. Columbia University Press, 1958.

[82] Y. Hahn and B. Lee. Identification of nine human-specific frameshift mutations by comparative analysis of the human and the chimpanzee genome sequences. *Bioinformatics*, 21(Suppl 1):i186–i194, 2005.

[83] P. Harrison and Z. Yu. Frame disruptions in human mRNA transcripts, and their relationship with splicing and protein structures. *BMC Genomics*, 8:371, 2007.

[84] S. Hayette, D. Dhermy, M.E. Dos Santos, M. Bozon, D. Drenckhahn, N. Alloisio, P. Texier, J. Delaunay, and L. Morlé. A deletional frameshift mutation in protein 4.2 gene (allele 4.2 Lisboa) associated with hereditary hemolytic anemia. *Blood*, 85(1):250, 1995.

[85] J. Hein and J. Støvlbæk. An algorithm combining DNA and protein alignment. *Journal of Theoretical Biology*, 167(2):169–174, 1994.

[86] S. Henikoff and J.G. Henikoff. Amino Acid Substitution Matrices from Protein Blocks. *Proc of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[87] P.G. Higgs and T.K. Attwood. *Bioinformatics and molecular evolution*, chapter 2. Blackwell, 2005.

[88] P.G. Higgs and T.K. Attwood. *Bioinformatics and molecular evolution*, chapter 3. Blackwell, 2005.

[89] C. Hoffmann, N. Minkah, J. Leipzig, G. Wang, M.Q. Arens, P. Tebas, and F.D. Bushman. DNA bar coding and pyrosequencing to identify rare HIV drug resistance mutations. *Nucleic Acids Research*, 35(13):e91, 2007.

[90] P.R. Hoffmann and M.J. Berry. Selenoprotein synthesis: a unique translational mechanism used by a diverse family of proteins. *Thyroid*, 15(8):769–775, 2005.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[91] S. Hoffmann, C. Otto, S. Kurtz, C. Sharma, P. Khaitovich, P.F. Stadler, and J. Hackermuller. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comp. Biol*, 5(9), 2009.

[92] N. Homer, B. Merriman, and S.F. Nelson. BFAST: an alignment tool for large scale genome resequencing. *PLoS ONE*, 4(11), 11 2009.

[93] N. Homer, B. Merriman, and S.F. Nelson. Local alignment of two-base encoded DNA sequence. *BMC bioinformatics*, 10(1):175, 2009.

[94] J. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. *The Theory of Machines and Computations*, pages 189–196, 1971.

[95] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, chapter 3: Regular expressions and languages. Pearson Addison Wesley, 2007.

[96] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, chapter 2: Finite automata. Pearson Addison Wesley, 2007.

[97] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, chapter 3: Properties of regular languages. Pearson Addison Wesley, 2007.

[98] S.Y. Hu, T. Liu, Z. Liu, E. Ledet, C. Velasco-Gonzalez, D.M. Mandal, and S. Koochekpour. Identification of a novel germline missense mutation of the androgen receptor in African American men with familial prostate cancer. *Asian Journal of Andrology*, 2010.

[99] L. Ilie and S. Ilie. Fast computation of good multiple spaced seeds. In *Proceedings of the 7th International Workshop in Algorithms in Bioinformatics (WABI), Philadelphia (USA)*, volume 4645 of *Lecture Notes in Bioinformatics*, pages 346–358. Springer, Sept. 2007.

[100] L. Ilie and S. Ilie. Multiple spaced seeds for homology search. *Bioinformatics*, 23(22):2969–2977, Sept. 2007.

[101] C.A. Janeway, P. Travers, M. Walport, and M.J. Shlomchik. *Immunobiology (5th ed.)*. Garland Science, 2005.

[102] N.C. Jones and P. Pevzner. *An introduction to bioinformatics algorithms*, chapter 6: Dynamic programming algorithms. the MIT Press, 2004.

[103] T.H. Jukes and S. Osawa. The genetic code in mitochondria and chloroplasts. *Cellular and Molecular Life Sciences*, 46(11):1117–1126, 1990.

[104] S. Karlin and S.F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences of the United States of America*, 87(6):2264–2268, 1990.

[105] U. Keich, M. Li, B. Ma, and J. Tromp. On spaced seeds for similarity search. *Discrete Applied Mathematics*, 138(3):253–263, 2004. (preliminary version in 2002).

[106] L. Knecht. Pairwise alignment with scoring on tuples. In *Combinatorial Pattern Matching*, pages 215–229. Springer, 1995.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[107] C. Kosiol, I. Holmes, and N. Goldman. An Empirical Codon Model for Protein Sequence Evolution. *Molecular Biology and Evolution*, 24(7):1464–1479, 2007.

[108] G. Kucherov, L. Noé, and M. Roytberg. A unifying framework for seed sensitivity and its application to subset seeds. *Algorithms in Bioinformatics*, pages 251–263, 2005.

[109] G. Kucherov, L. Noé, and M. Roytberg. Multiseed lossless filtration. *IEEE Transactions on Computational Biology and Bioinformatics*, 2(1):51–61, 2005.

[110] G. Kucherov, L. Noé, and M. Roytberg. A unifying framework for seed sensitivity and its application to subset seeds. *J Bioinform Comput Biol*, 4(2):553–570, 2006.

[111] G. Kucherov, L. Noé, and M. Roytberg. Subset seed automaton. In *Proceedings of the 12th International Conference on Implementation and Application of Automata (CIAA)*, volume 4783 of *LNCS*, pages 180–191. Springer, 2007.

[112] G. Kucherov, L. Noé, and M. Roytberg. Iedera: subset seed design tool. http://bioinfo.lifl.fr/yass/iedera, 2010.

[113] G. Kucherov, T. Pinhas, and M. Ziv-Ukelson. Regular Language Constrained Sequence Alignment Revisited. In *Proceedings of IWOCA 2010*, LNCS, page (to appear). Springer, 2010.

[114] T.A. Kunkel. DNA Replication Fidelity. *Journal of Biological Chemistry*, 279(17):16895–16898, 2004.

[115] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3), 2009.

[116] J.D. Lee, S.C. Kim, Y.W. Koh, H.J. Lee, S.Y. Choi, and U.K. Kim. A Novel Frameshift Mutation in the EYA1 Gene in a Korean Family with Branchio-Oto-Renal Syndrome. *Annals of Clinical & Laboratory Science*, 39(3):303, 2009.

[117] J. Leluk. A new algorithm for analysis of the homology in protein primary structure. *Computers and Chemistry*, 22(1):123–131, 1998.

[118] J. Leluk. A non-statistical approach to protein mutational variability. *BioSystems*, 56(2-3):83–93, 2000.

[119] M.Y. Leung, B.E. Blaisdell, C. Burge, and S. Karlin. An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *Journal of Molecular Biology*, 221(4):1367–1378, 1991.

[120] P.A. Levene. The structure of Yeast nucleic acid. *Journal of Biological Chemistry*, 40(2):415–424, 1919.

[121] T.J. Ley, E.R. Mardis, L. Ding, B. Fulton, M.D. McLellan, K. Chen, D. Dooling, B.H. Dunford-Shore, S. McGrath, M. Hickenbotham, et al. DNA sequencing of a cytogenetically normal acute myeloid leukaemia genome. *Nature*, 456(7218):66–72, 2008.

[122] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[123] H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18:1851–1858, 2008.

[124] M. Li, B.G. Fry, and R.M. Kini. Eggs-only diet: Its implications for the toxin profile changes and ecology of the marbled sea snake (aipysurus eydouxii). *Journal of Molecular Evolution*, 60(1):81–89, 2005.

[125] M. Li, B. Ma, D. Kisman, and J. Tromp. PatternHunter II: Highly sensitive and fast homology search. *Journal of Bioinformatics and Computational Biology*, 2(3):417–439, 2004. (earlier version in GIW 2003).

[126] R. Li, C. Yu, Y. Li, T. Lam, S. Yiu, K. Kristiansen, and J. Wang. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15):1966–1967, 2009.

[127] P. Licznar, C. Bertrand, I. Canal, M.-F. Prère, and O. Fayet. Genetic variability of the frameshift region in IS911 transposable elements from escherichia coli clinical isolates. *FEMS Microbiology Letters*, 218(2):231–237, 2006.

[128] H. Lin, Z. Zhang, M.Q. Zhang, B. Ma, and M. Li. ZOOM! zillions of oligos mapped. *Bioinformatics*, 24(21):2431–2437, 2008.

[129] P. Lio and N. Goldman. Models of Molecular Evolution and Phylogeny. *Genome Research*, 8(12):1233–1244, 1998.

[130] D.J. Lipman and W.R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, 1985.

[131] H. Lodish, A. Berk, L.S. Zipursky, P. Matsudaira, D. Baltimore, and J. Darnell. *Molecular Cell Biology*, chapter 3.1 Hierarchical Structure of Proteins. W. H. Freeman and Company, 2000.

[132] H. Lodish, A. Berk, L.S. Zipursky, P. Matsudaira, D. Baltimore, and J. Darnell. *Molecular Cell Biology*, chapter 12.4 DNA Damage and Repair and Their Role in Carcinogenesis. W. H. Freeman and Company, 2000.

[133] B. Ma, J. Tromp, and M. Li. PatternHunter: Faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

[134] B. Ma and H. Yao. Seed optimization for i.i.d. similarities is no easier than optimal Golomb ruler design. *Information Processing Letters*, 109:1120–1124, 2009.

[135] D. Mak, Y. Gelfand, and G. Benson. Indel seeds for homology search. *Bioinformatics*, 22(14):e341, 2006.

[136] E.R. Mardis. The impact of next-generation sequencing technology on genetics. *Trends in Genetics*, 24(3):133–141, 2008.

[137] M. Margulies, M. Egholm, W.E. Altman, S. Attiya, J.S. Bader, L.A. Bemben, J. Berka, M.S. Braverman, Y.J. Chen, Z. Chen, et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, 2005.

[138] J.H. Matthaei, O.W. Jones, R.G. Martin, and M.W. Nirenberg. Characteristics and composition of RNA coding units. *Proceedings of the National Academy of Sciences of the United States of America*, 48(4):666–677, 1962.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[139] A.M. Maxam and W. Gilbert. A new method for sequencing DNA. *Proc. Natl. Acd. Sci. USA*, 74(2):560–564, 1977.

[140] B. McClintock. The origin and behavior of mutable loci in maize. *Proceedings of the National Academy of Sciences of the United States of America*, 36(6):344–355, 1950.

[141] S.D. McCulloch and T.A. Kunkel. The fidelity of DNA synthesis by eukaryotic replicative and translasion synthesis polymerases. *Cell Research*, 18(1):148–161, 2008.

[142] K.J. McKernan, A.P. Blanchard, L. Kotler, and G. Costa. Reagents, methods, and libraries for bead-based sequencing, 2006. WO Patent WO/2006/084,132.

[143] M.H. Meisler and J.A. Kearney. Sodium channel mutations in epilepsy and other neurological disorders. *Journal of Clinical Investigation*, 115(8):2010–2017, 2005.

[144] M.L. Metzker. Sequencing technologies – the next generation. *Nature Reviews Genetics*, 11(1):31–46, 2009.

[145] A. Meyer and Y. Van de Peer. 'Natural selection merely modified while redundancy created'–Susumu Ohno's idea of the evolutionary importance of gene and genome duplications. *Journal of Structural and Functional Genomics*, 3(1):7–9, 2003.

[146] E.F. Moore. Gedanken-experiments on sequential machines. *Automata studies*, 34:129–153, 1956.

[147] A. Moreira and A. Maass. TIP: protein backtranslation aided by genetic algorithms. *Bioinformatics*, 20(13):2148–2149, 2004.

[148] K.B. Mullis, F. Ferré, and R.A. Gibbs. *The polymerase chain reaction*. Springer Science & Business, 1994.

[149] H. Nan, T. Niu, D.J. Hunter, and J. Han. Missense Polymorphisms in Matrix Metalloproteinase Genes and Skin Cancer Risk. *Cancer Epidemiology Biomarkers & Prevention*, 17(12):3551, 2008.

[150] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–53, 1970.

[151] F. Nicolas and E. Rivals. Hardness of optimal spaced seed design. *Journal of Computer and System Sciences*, 74(5):831–849, 2008.

[152] M.W. Nirenberg. The genetic code. *Les Prix Nobel en*, pages 221–241, 1968.

[153] M.W. Nirenberg, P. Leder, M. Bernfield, R. Brimacombe, J. Trupin, F. Rottman, and C. O'Neal. RNA codewords and protein synthesis, VII. On the general nature of the RNA code. *Proceedings of the National Academy of Sciences of the United States of America*, 53(5):1161–1168, 1965.

[154] L. Noé, M. Gîrdea, and G. Kucherov. Designing efficient spaced seeds for SOLiD read mapping. *Advances in Bioinformatics*, July 2010.

[155] L. Noé, M. Gîrdea, and G. Kucherov. Seed design framework for mapping SOLiD reads. In B. Berger, editor, *Proceedings of the 14th Annual International Conference on Research in Computational Molecular Biology (RECOMB), April 25-28, 2010, Lisbon (Portugal)*, volume 6044 of *Lecture Notes in Computer Science*, pages 384–396. Springer, April 2010.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[156] L. Noé and G. Kucherov. Improved hit criteria for DNA local alignment. *BMC Bioinformatics*, 5(149), october 2004.

[157] L. Noé and G. Kucherov. YASS: enhancing the sensitivity of DNA similarity search. *Nucleic Acids Research*, 33 (web-server issue)(suppl_2):W540–W543, Jul 2005.

[158] S. Ohno. *Evolution by gene duplication.* New York (EUA). Springer-Verlag, 1970.

[159] K. Okamura, L. Feuk, T. Marquès-Bonet, A. Navarro, and S.W. Scherer. Frequent appearance of novel protein-coding sequences by frameshift translation. *Genomics*, 88(6):690–697, 2006.

[160] R.J. Olds, D.A. Lane, G. Finazzi, T. Barbui, and S.L. Thein. A frameshift mutation leading to type 1 antithrombin deficiency and thrombosis. *Blood*, 76(11):2182, 1990.

[161] M. Olivier, M. Hollstein, and P. Hainaut. TP53 mutations in human cancers: origins, consequences, and clinical use. *Cold Spring Harbor Perspectives in Biology*, 2(1), 2010.

[162] R. Olsen, R. Bundschuh, and T. Hwa. Rapid assessment of extremal statistics for gapped local alignment. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 211–222. AAAI press, 1999.

[163] B.D. Ondov, A. Varadarajan, K.D. Passalacqua, and N.H. Bergman. Efficient mapping of Applied Biosystems SOLiD sequence data to a reference genome for functional genomic applications. *Bioinformatics*, 24(23):2776–2777, 2008.

[164] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448, 1988.

[165] H.E. Peckham, S.F. McLaughlin, J.N. Ni, M.D. Rhodes, J.A. Malek, K.J. McKernan, and A.P. Blanchard. SOLiD^{TM} Sequencing and 2-Base Encoding, 2007.

[166] C. Pedersen, R. Lyngsø, and J. Hein. Comparison of coding DNA. In *Combinatorial Pattern Matching*, pages 153–173. Springer, 1998.

[167] M. Pellegrini and T.O. Yeates. Searching for Frameshift Evolutionary Relationships Between Protein Sequence Families. *Proteins*, 37:278–283, 1999.

[168] H. Peltola, H. Soderlund, and E. Ukkonen. Algorithms for the search of amino acid patterns in nucleic acid sequences. *Nucleic acids research*, 14(1):99–107, 1986.

[169] N. Pourmand, M. Karhanek, H.H.J. Persson, C.D. Webb, T.H. Lee, A. Zahradníková, and R.W. Davis. Direct electrical detection of DNA synthesis. *Proceedings of the National Academy of Sciences*, 103(17):6466–6470, 2006.

[170] J.M. Prober, G.L. Trainor, R.J. Dam, F.W. Hobbs, C.W. Robertson, R.J. Zagursky, A.J. Cocuzza, M.A. Jensen, and K. Baumeister. A System for Rapid DNA Sequencing with Fluorescent Chain-Terminating Dideoxynucleotides. *Science*, 238:336–341, 1987.

[171] K. Prufer, U. Stenzel, M. Dannemann, R.E. Green, M. Lachmann, and J. Kelso. PatMaN: rapid alignment of short sequences to large databases. *Bioinformatics*, 24(13):1530–1531, 2008.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[172] J. Raes and Y. Van de Peer. Functional divergence of proteins through frameshift mutations. *Trends in Genetics*, 21(8):428–431, 2005.

[173] E. Rivals, L. Salmela, P. Kiiskinen, P. Kalsi, and J. Tarhio. MPSCAN: Fast localisation of multiple reads in genomes. In *Proceedings of the 9th international Workshop on Algorithms in Bioinformatics (WABI)*, volume 5724 of *LNCS*, pages 246–260. Springer, 2009.

[174] G. Rizk and D. Lavenier. GASSST: Global Alignment Short Sequence Search Tool. *Bioinformatics*, 2010.

[175] A. Rojas, S. Garcia-Vallvé, M.A. Montero, L. Arola, and A. Romeu. Frameshift mutation events in beta-glucosidases. *Gene*, 314:191–199, 2003.

[176] M. Ronaghi, S. Karamohamed, B. Pettersson, M. Uhlen, and P. Nyrén. Real-time DNA sequencing using detection of pyrophosphate release. *Analytical biochemistry*, 242(1):84–89, 1996.

[177] M. Roytberg, A. Gambin, L. Noé, S. Lasota, E. Furletova, E. Szczurek, and G. Kucherov. On subset seeds for protein alignment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(3):483–494, 2009.

[178] E.M. Rubin, S. Lucas, P. Richardson, D. Rokhsar, and L. Pennacchio. Finishing the euchromatic sequence of the human genome. *Nature*, 431, 2004.

[179] S.M. Rumble, P. Lacroute, A.V. Dalca, M. Fiume, A. Sidow, and M. Brudno. SHRiMP: Accurate mapping of short color-space reads. *PLoS Comp. Biol*, 5(5), 2009.

[180] R.K. Saiki, D.H. Gelfand, S. Stoffel, S.J. Scharf, R. Higuchi, G.T. Horn, K.B. Mullis, and H.A. Erlich. Primer-directed enzymatic amplification of DNA with a thermostable DNA polymerase. *Science*, 239(4839):487–491, 1988.

[181] F. Sanger and A.R. Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase* 1. *Journal of Molecular Biology*, 94(3):441–446, 1975.

[182] A. Schneider, G.M. Cannarozzi, and G.H. Gonnet. Empirical codon substitution matrix. *BMC bioinformatics*, 6(1):134, 2005.

[183] S.C. Schuster. Next-generation sequencing transforms today's biology. *Nature Methods*, 5(1):16–18, 2007.

[184] S. Schwartz, W.J. Kent, A. Smit, Z. Zhang, R. Baertsch, R.C. Hardison, D. Haussler, and W. Miller. Human–mouse alignments with BLASTZ. *Genome Research*, 13(1):103–107, 2003.

[185] J. Shendure and H. Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145, 2008.

[186] J. Shendure, G.J. Porreca, N.B. Reppas, X. Lin, J.P. McCutcheon, A.M. Rosenbaum, M.D. Wang, K. Zhang, R.D. Mitra, and G.M. Church. Accurate Multiplex Polony Sequencing of an Evolved Bacterial Genome. *Science*, 309(5741):1728–1732, 2005.

[187] J.C. Shepherd. Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification. *Proceedings of the National Academy of Sciences*, 78(3):1596–1600, 1981.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[188] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Bwl*, 147:195–197, 1981.

[189] D. Söll and U. RajBhandary. *tRNA: structure, biosynthesis, and function*, chapter 11: Byosynthesis and Function of Modified Nucleosides. American Society for Microbiology, 1995.

[190] H.H. Stedman, B.W. Kozyak, A. Nelson, D.M. Thesier, L.T. Su, D.W. Low, C.R. Bridges, J.B. Shrager, N. Minugh-Purvis, and M.A. Mitchell. Myosin gene mutation correlates with anatomical changes in the human lineage. *Nature*, 428(6981):415–418, 2004.

[191] D. Stoddart, A.J. Heron, E. Mikhailova, G. Maglia, and H. Bayley. Single-nucleotide discrimination in immobilized DNA oligonucleotides with a biological nanopore. *Proceedings of the National Academy of Sciences*, 106(19):7702–7707, 2009.

[192] D. Stoddart, G. Maglia, , E. Mikhailova, A.J. Heron, and H. Bayley. Multiple Base-Recognition Sites in a Biological Nanopore: Two Heads are Better than One. *Angewandte Chemie*, 49(3):556–559, 2010.

[193] P. Stothard. The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences. *Biotechniques*, 28(6):1102–1104, 2000.

[194] G. Streisinger, Y. Okada, J. Emrich, J. Newton, A. Tsugita, E. Terzaghi, and M. Inouye. Frameshift mutations and the genetic code. In *Cold Spring Harbor Symp. Quant. Biol*, volume 31, pages 77–84, 1966.

[195] M. Strömberg and W.-P. Lee. MOSAIK read alignment and assembly program. http://bioinformatics.bc.edu/marthlab/Mosaik, 2009.

[196] M.J. Stuart and R.L. Nagel. Sickle-cell disease. *The Lancet*, 364(9442):1343–1360, 2004.

[197] Y. Sun and J. Buhler. Designing multiple simultaneous seeds for DNA similarity search. *Journal of Computational Biology*, 12(6):847–861, 2005.

[198] M. Suyama, D. Torrents, and P. Bork. PAL2NAL: robust conversion of protein sequence alignments into the corresponding codon alignments. *Nucleic Acids Research*, 34 (Web Server issue):W609–W612, 2006.

[199] Oxford Nanopore Technologies. DNA Sequencing.

[200] Ion Torrent. Technology: The Simplest Sequencing Chemistry. http://www.iontorrent.com/the-simplest-sequencing-chemistry/.

[201] C. Touriol, S. Bornes, S. Bonnal, S. Audigier, H. Prats, A.C. Prats, and S. Vagner. Generation of protein isoform diversity by alternative initiation of translation at non-AUG codons. *Biology of the Cell*, 95(3-4):169–178, 2003.

[202] G. Turcatti, A. Romieu, M. Fedurco, and A.P. Tairi. A new class of cleavable fluorescent nucleotides: synthesis and optimization as reversible terminators for DNA sequencing by synthesis. *Nucleic Acids Research*, 36(4):e25, 2008.

[203] E. Vennemeyer, S. Jankowski, and K. Hunkapiller. P226-S Direct Sequencing Quality Control: A Novel Software Approach to Reducing Variant Review Time and Labor. *Journal of Biomolecular Techniques: JBT*, 18(1):78, 2007.

http://doc.univ-lille1.fr

Thèse de Marta Girdéa, Lille 1, 2010

[204] A. Vignal, D. Milan, M. SanCristobal, and A. Eggen. A review on SNP and other types of molecular markers and their use in animal genetics. *Genetics Selection Evolution*, 34(3):275–305, 2002.

[205] K.V. Voelkerding, S.A. Dames, and J.D. Durtschi. Next-generation sequencing: from basic research to diagnostics. *Clinical chemistry*, 55(4):641, 2009.

[206] C. Wang, Y. Mitsuya, B. Gharizadeh, M. Ronaghi, and R.W. Shafer. Characterization of mutation spectra with ultra-deep pyrosequencing: application to HIV-1 drug resistance. *Genome research*, 17(8):1195, 2007.

[207] J.D. Watson and F.H.C. Crick. A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[208] D. Weese, A.K. Emde, T. Rausch, A. Döring, and K. Reinert. RazerS–fast read mapping with sensitivity control. *Genome Research*, 19(9):1646–1654, 2009.

[209] R. Wernersson and A.G. Pedersen. RevTrans: Multiple alignment of coding DNA from aligned amino acid sequences. *Nucleic Acids Research*, 31(13):3537–3539, 2003.

[210] J. Xu, D.G. Brown, M. Li, and B. Ma. Optimizing multiple spaced seeds for homology search. *Journal of Computational Biology*, 13(7):1355–1368, 2006. (earlier version in CPM 2004).

[211] F.I. Yamamoto. Molecular genetics of the ABO histo-blood group system. *Vox sanguinis*, 69(1):1–7, 1995.

[212] Y. Ye and H. Tang. Dynamic programming algorithms for biological sequence and structure comparison. *Bioinformatics algorithms: techniques and applications*, pages 9–28, 2008.

[213] J. Zhang. Evolution by gene duplication: an update. *Trends in Ecology & Evolution*, 18(6):292–298, 2003.

[214] Z. Zhang, W.R. Pearson, and W. Miller. Aligning a DNA sequence with a protein sequence. *Journal of Computational Biology*, 4(3):339–349, 1997.

[215] L. Zhou, I. Mihai, and L. Florea. Spaced seeds for cross-species CDNA-to-genome sequence alignment. *Communications in Information and Systems*, 10(2):115–136, 2010.

[216] L. Zhou, J. Stanton, and L. Florea. Universal seeds for cDNA-to-genome comparison. *BMC Bioinformatics*, 9(36), 2008.

[217] S. Züchner, I.V. Mersiyanova, M. Muglia, N. Bissar-Tadmouri, J. Rochelle, E.L. Dadali, M. Zappia, E. Nelis, A. Patitucci, J. Senderek, et al. Mutations in the mitochondrial GTPase mitofusin 2 cause Charcot-Marie-Tooth neuropathy type 2A. *Nature genetics*, 36(5):449–451, 2004.

http://doc.univ-lille1.fr