Recherche et annotation des structures de CRISPR dans l'ensemble des génomes procaryotes

Christophe Vroland, Catherine Belleannée, Jacques Nicolas

Publications Internes de l'IRISA
ISSN : 2102-6327
PI 1986 – 21 novembre 2011

Research and annotation of CRISPR structures in all prokaryotic genomes

Christophe Vroland[*] , Catherine Belleannée[**], Jacques Nicolas[**]
Catherine.Belleannee@irisa.fr, Jacques.Nicolas@irisa.fr

Abstract: CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) are structures of repeated units widely present in prokaryotic genomes but absent in Eukarya genomes. We previously implemented a relational databasis continuously fed by an automatic update, called Crispi, to detect and bring together all the CRISPR. The purpose here is to enrich the databasis by adding the annotation of palindromic structures identified in CRISPR elements. The identification of these elements (stem-loop structures) was performed by means of a grammatical modeling, using the Logol pattern-matching software. The definition of a stem-loop model well-suited to the CRISPR sequences set up a sensitive part of this work.

Key-words: Bioinformatics, genome analysis, CRISPR, stem-loop identification

---

Recherche et annotation des structures de CRISPR dans l'ensemble des génomes procaryotes

Résumé : *Les structures de CRISPR sont des suites d'unités répétées largement présentes dans les génomes procaryotes mais absentes des génomes eucaryotes. Nous avons déjà réalisé une base de données relationnelle, Crispi, rassemblant l'ensemble des CRISPR, et alimentée en permanence par une mise à jour automatique. Il s'agit ici d'enrichir la base en ajoutant l'annotation des structures présentes dans les éléments CRISPR. L'identification de ces structures palindromiques (tige-boucles) a été réalisée à l'aide d'un modèle grammatical, en utilisant le logiciel de pattern-matching Logol. La définition du modèle de tige-boucle adapté aux CRISPRS a constitué un des points sensibles de ces travaux.*

Mots clés : *Bioinformatique, analyse de génomes,CRISPR, identification de structure en tige-boucle*

Avec le soutien de la fondation Rennes1
[*] En stage M2-CCI dans Symbiose en 2011-2012
[**] Equipe Symbiose, Irisa/Inria/Université de Rennes1

1 Introduction

Les structures de CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) sont des suites d'unités répétées largement présentes dans les génomes bactériens et d'archées mais absentes des génomes eucaryotes (9). Ces structures jouent un rôle important dans l'immunité microbienne contre les virus en captant une petite partie de leurs génomes et en utilisant un mécanisme d'interférence ARN qu'on peut rapprocher du mécanisme d'interférence chez les eucaryotes. Dans les biotechnologies, la connaissance des CRISPR permet d'envisager une meilleure maîtrise du rendement des bioréacteurs. L'évolution rapide des séquences de CRISPR est également utile pour accéder à la micro-évolution des espèces et permettre leur génotypage en épidémiologie. Malgré des études récentes nombreuses, une vue complète et précise de l'ensemble des CRISPR et des structures associées dans les génomes séquencés reste à construire, ce qui constitue un frein à l'établissement d'hypothèses sur le rôle et la dynamique de ces éléments.

Nous décrivons ici une contribution à la caractérisation de la nature palindromique des CRISPR. Dans l'équipe Symbiose (*Sy*stèmes et *m*odèles *bio*logiques, bio-informatique et *sé*quences) à Rennes, nous avons réalisé une base de données relationnelle sur le thème des CRISPR, la base CRISPI (http://crispi.genouest.org/)(14), alimentée en permanence par un système de mise à jour automatique. Le but des travaux décrits dans ce document est d'enrichir la base d'un nouveau module. Il concerne la détection de structures palindromiques (les « tige-boucles ») sur les éléments CRISP et leur annotation automatique dans la base CRISPI, ces structures secondaires étant supposées avoir un rôle déterminant dans la fonctionnalité des CRISPR. La finalité de l'ensemble de ces travaux est de faire de CRISPI une base de référence dans le domaine.

Les travaux présentés ici[1] s'articulent en deux étapes.
La première consiste à définir un modèle de tige-boucle conforme à la réalité des CRISPR. Pour établir le modèle ayant la meilleure pertinence statistique, le modèle est ici confronté à trois jeux de séquences, les séquences cibles (les « repeats » de CRISP), des séquences aléatoires et des séquences biologiques non cibles (les « spacers » de CRISPR) afin de tenter d'optimiser la spécificité du modèle.
La deuxième étape consiste à mettre en œuvre l'annotation de la base CRISPI avec ces informations de structure. Il s'agit dans un premier temps de réaliser la recherche de telles tige-boucles dans l'ensemble des CRISPR. Notre approche de « pattern-matching » est basée sur une modélisation unifiée par grammaire formelle des éléments à rechercher. Dans Symbiose, nous avons précédemment défini un langage de description de haut niveau utilisant les grammaires à variables de chaînes (15) comme construction fondamentale, et mis en œuvre l'analyseur associé (outils Stan(13) puis Logol (3)). Les motifs de tige-boucles seront pour l'essentiel décrits en Logol et recherchés dans les séquences biologiques avec l'analyseur associé.
Il s'agit ensuite d'intégrer à la base CRISPI l'annotation des structures secondaires détectées. Cela amène à modifier à la fois le schéma de la base relationnelle et le site web associé, afin de permettre l'affichage des données de structures au niveau de l'interface graphique.

Dans ce document, une place importante est faite à la partie sensible qu'est la conception du modèle. Par ailleurs un certain nombre de considérations techniques sont présentées dans ce document, les plus détaillées étant reportées en annexe..

1 Une grande partie de ces travaux a été effectuée dans le cadre d'un stage de Master2 CCI de l'Université de Rennes1 en 2011-2012.

2 Contexte de l'étude

2.1 Un mécanisme de défense des procaryotes : les CRISPR

Les CRISPR ou « Clustered regularly interspaced short palindromic repeats » (traduction littérale : courtes répétitions en palindrome regroupées et régulièrement espacées) sont des structures répétées retrouvées dans les génomes des bactéries et des archées. Actuellement, les définitions en restent assez informelles et reposent sur des observations empiriques. Les CRISPR sont « composés de la répétition exacte de séquences (les « *repeats* ») de 24 à 48 bases séparées par des « *spacers* »uniques d'une taille similaire » (12)(10)(11) (figure 1). Bien que cette définition postule que les séquences répétées (les repeats) d'un même CRISPR doivent être identiques, de légères variations dans le train de répétitions sont observées. De plus, il arrive parfois que les spacers soient dupliqués à l'intérieur même d'un CRISPR, voire dans deux CRISPR différents sur un même génome. Ainsi une autre définition du CRISPR pourrait être « une suite périodiquement espacée d'unités (au moins quatre repeats) qui ne soit pas une répétition en tandem »(14).

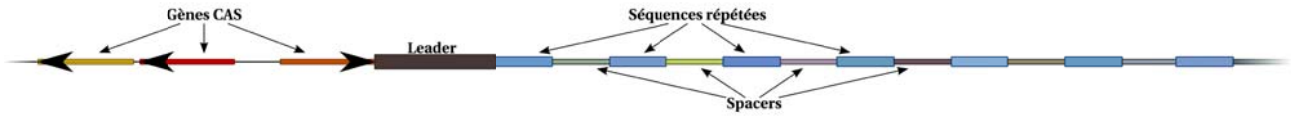


figure 1: schéma d'un CRISPR

Le CRISPR est composé de la répétition d'une séquence (le 'repeat') espacée par des séquences uniques (les 'spacers'). Le nombre de répétitions peut aller jusqu'à 361 pour *Isosphaera pallida* ATCC 43644 (selon la base de données CRISPI). Un famille de gènes, les gènes *cas*, est souvent retrouvée à proximité des CRISPR. Ces gènes joueraient un rôle dans l'insertion de nouvelles unités et dans l'action du CRISPR.

Les CRISPR ont été découverts pour la première fois en 1987 chez *E. coli*. Depuis la recherche de ces structures *in silico* dans les génomes séquencés ont révélé que les CRISPR sont retrouvés dans environ 90% des archées (82.83% selon CRISPI) et 40% des bactéries (56.38% selon CRISPI) (9).
L'observation des séquences répétées, que l'on nomme « repeats », laisse à penser qu'un certain nombre d'entre elles sont capables de réaliser des structures secondaires du type « tige-boucle » (cf section 2.3).
Pour les séquences séparant deux repeats, que l'on nomme « spacers », il a été observé qu'elles sont similaires à des séquences provenant de virus de bactéries[2] mais aussi de plasmides[3]. Cependant des études poussées sont assez difficiles à réaliser car l'ensemble des génomes potentiellement concernés n'a pas été séquencé.
Ces structures sont souvent adjacentes à un ensemble de gènes codant pour des protéines aux rôles divers (nucléases, hélicases, polymérases et des protéines fixant les polynucléotides.). Ces gènes sont dit gènes « *cas* » pour « CRISPR Associated ». Leur rôle exact n'est pas encore très bien connu.
Il a été démontré que les CRISPR sont impliqués dans la résistance des procaryotes contre les virus. En effet, il a été observé chez *Streptococcus thermophilus* (intervenant dans la fabrication du yaourt

2 nommé phage/bactériophage : ces virus ne touchent que les bactéries et les archées.
3 Les plasmides sont des séquences d'ADN mobile, généralement circulaires qui viennent compléter l'ADN des procaryotes (et de très rares eucaryotes). Il peuvent apporter diverses fonctions : résistance aux antibiotiques, nouvelles voies métaboliques...mais ceux-ci peuvent avoir des comportements proches de parasites.

et de certains fromages) mis en contact avec des virus que la machinerie est capable d'acquérir de nouvelles unités dans leur CRISPR. Les nouveaux spacers ainsi acquis sont identiques à des portions du génome des virus. Les bactéries modifiées sont devenues résistantes à ces virus(2)(4).

2.2 Les bases de données de CRISPR

2.2.1 CRISPI

L'équipe Symbiose a conçu une base de données, CRISPI, afin de regrouper l'ensemble des CRISPR sur tous les génomes des procaryotes(14). Cette base de données est accessible à travers un site internet (http://crispi.genouest.org/). Au moment de la rédaction de ce document, elle contient les données CRISPR de 118 génomes d'archées et 1467 génomes de bactéries, pour un total de 57484 repeats (cf figure 2).

Elle est remplie à l'aide d'un algorithme de recherche de CRISPR développé au sein de l'équipe. Une routine de mise à jour permet de la déclencher pour nourrir continuellement la base. Elle prend comme données d'entrée les génomes accessibles à partir des serveurs du NCBI (ftp.ncbi.nih.gov/genomes/Bacteria/). Seuls les génomes modifiés ou nouvellement ajoutés sont mis à jour. La routine est implémentée via l'environnement générique BIOMAJ (http://biomaj.genouest.org/)(5).
Le site internet propose différentes fonctionnalités à l'utilisateur. La recherche de CRISPR peut être réalisée sur plusieurs critères : le nom du génome, son numéro d'accession ou encore en parcourant un arbre taxonomique. Il affiche les résultats ainsi que diverses informations à propos du génome choisi.
Le site propose aussi de confronter un jeu de séquences donné par l'utilisateur à l'ensemble des spacers de la base de données à l'aide logiciel BLAST(1). Il permet aussi de rechercher les CRISPR contenus dans une séquence fournie par l'utilisateur, via l'algorithme de recherche.

Dans la base de données, chaque CRISPR possède une séquence consensus. Cette séquence est sen sée représenter la « moyenne » de tous les repeats du CRISPR. Du fait que les repeats sont très

Consult CRISPR Database

Database Summary

	Genomes analysed	Species analysed	CRISPRs found	Genomes with CRISPR	Species with CRISPR	Units found	Spacers found	Putative CAS genes
Archaea	118	102	399	94 (79.66%)	85 (83.33%)	11202	10803	1466
Bacteria	1467	939	1650	690 (47.03%)	529 (56.34%)	46282	44632	5503

Last Update = 2011-12-05 12:54:05

Database Highlights

Genomes with maximal	number of CRISPR	number of units in a CRISPR	number of CAS genes	spacer size	repeat size
Archaea	Methanocaldococcus sp. FS406-22 chromosome (18)	Sulfolobus tokodaii str. 7 chromosome (458)	Pyrococcus yayanosii CH1 chromosome (47)	Methanococcus vannielii SB chromosome (488)	Methanospirillum hungatei JF-1 chromosome (39)
Bacteria	Thermomonospora curvata DSM 43183 chromosome (13)	Haliangium ochraceum DSM 14365 chromosome (743)	Rhodospirillum rubrum ATCC 11170 chromosome (44)	Xylella fastidiosa M23 chromosome (574)	Shewanella putrefaciens CN-32 chromosome (92)

figure 2: Données statistiques de la base CRISPI

fortement conservés au sein d'un CRISPR, les séquences consensus sont pour la plupart très similaires aux repeats. Le schéma relationnel de la base est visible (partie en noir) en figure 7.

2.2.2 CRISPRDB

CRISPRdb(7) est une base de données concurrente, répertoriant l'ensemble des CRISPR à l'aide de CRISPRfinder(6). La base est mise à jour automatiquement tous les mois avec les génomes modifiés ou nouvellement ajoutés. La base est aussi associée à un site internet, proposant également un set de fonctionnalités et d'outils.

2.3 Structures secondaires dans les séquences nucléiques

2.3.1 Définition

Dans une séquence nucléique, la structure secondaire décrit les zones d'hybridation entre sous-séquences complémentaires sur une même séquence.
Les structures secondaires sont fréquemment observées dans les ARN. Elles correspondent à des régions qui sont des séquences répétées inversées complémentaires. Ces sous-séquences vont naturellement s'apparier.
Les bases s'associent par paires avec leurs bases complémentaires (figure 3A).
Les paires dites canoniques, ou paires Watson-Crick, sont A-T (ou A-U dans l'ARN) et G-C. Les liaisons G-C sont celles qui possèdent l'énergie la plus basse. Une zone avec fort taux de paires G-C forme une zone d'appariement plus difficile à défaire qu'une zone avec un fort taux de paires A-T. Dans une moindre mesure, les bases azotées sont capables de réaliser d'autres appariements que ceux de Watson et Crick. Parmi ces appariements « bancals » ou *wobble pairing,* l'appariement G-U est très courant dans le repliements de l'ARN.
Une 'tige-boucle' (stemloop en anglais), ou structure en épingle à cheveux, est la structure secondaire la plus élémentaire (figure 3C). Elle se forme pour des séquence w. u. -w où w et -w sont deux séquences complémentaires inversées s'appariant pour former la *tige* et u est une séquence formant la *boucle*, sans appariement. Il existe des structures secondaires plus complexes, telles que le 'pseudo-nœud' (figure 6C) qui correspond à deux tige-boucles imbriquées, ou d'autres conformations incluant toutes sortes de boucles.

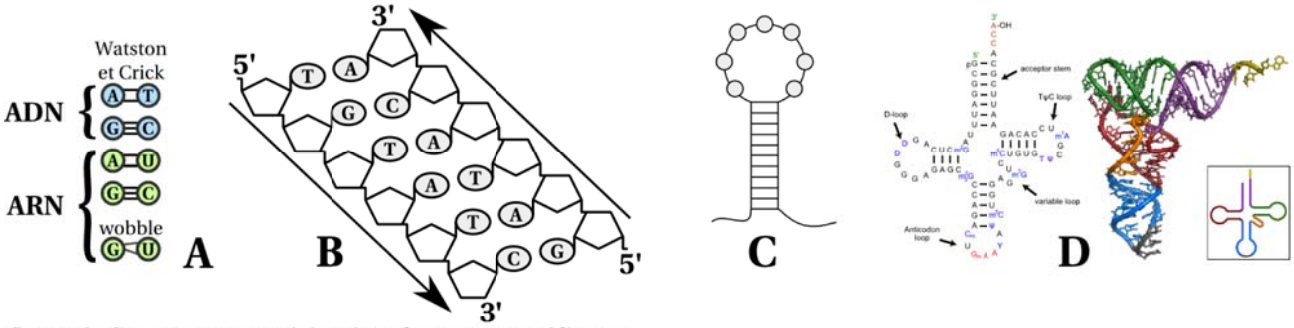


figure 3: Structure secondaire des séquences nucléiques

Les bases s'associent avec leurs bases complémentaires. Dans l'ADN, les paires sont A-T et G-C alors que dans l'ARN la base T est remplacée par la base U. En plus de ces appariements canoniques, il en existe d'autres appelés « wobble pairing » comme l'appariement G-U (**A**). Un brin s'hybride avec un brin complémentaire inversé (**B**). Deux régions complémentaires inversées sur un même brin qui s'hybrident forment une structure secondaire élémentaire : la tige-boucle (**C**). Structures en « feuille de trèfle » des ARN de transfert (**D**).

D'une façon générale, la présence de structures secondaires dans l'ARN est le signe que cette portion d'ARN est potentiellement impliquée dans un mécanisme cellulaire au travers d'interactions avec d'autres molécules. Il est donc important de rechercher ce type de structures.

2.3.2 Identification des structures secondaires par calcul d'énergie libre

A priori, les règles d'appariement des acides nucléiques sont simples et bien définies (dans l'ARN : A-U,C-G voire G-U). Il n'est cependant pas si aisé de calculer les structures secondaires à partir des séquences car la combinatoire d'appariement peut être très grande. Pour ces raisons de complexité, les modèles de calcul délaissent souvent les pseudo-nœuds. Parmi ces modèles, le modèle de Zuker *et al.* (17) est couramment utilisé. Il calcule la structure secondaire de plus faible énergie libre (plus l'énergie d'une structure est basse, plus celle-ci est stable, favorisant ainsi les liaisons C-G).
Ce calcul prend en compte des données expérimentales. Le modèle se base sur le fait que l'énergie libre des nucléotides est dépendante de la position de ces derniers dans la structure. Il suppose de plus que l'énergie libre est additive. Nous indiquons ici quelques détails du calcul.
Soit S_{ij} une structure secondaire allant du nucléotide d'indice i au nucléotide d'indice j. L'algorithme calcule deux matrices v et w. w(i,j) est l'énergie minimale de toutes les structures S_{ij} possibles. v(i,j) est l'énergie minimale des structures S_{ij} possibles où i et j sont appariés. Si i et j ne peuvent pas s'apparier, $v(i,j) = +\infty$.
w(i,j) et v(i,j) sont calculées récursivement jusqu'à ce que j-i = 5, c'est-à-dire jusqu'aux sous séquences de taille 5.
v(i,j) est calculée avec la formule suivante :

$$v(i,j)=min\begin{pmatrix} {}^{1}fh(i,j) \\ {}^{2}\forall i',\forall j', i<i'<j'<j, min\left(fl(i,j,i',j')+v(i',j')\right) \\ {}^{3}\forall i', i+1<i'<j-2, min\left(p(i,j,w(i+1,i')+w(i'+1,j-1))\right) \end{pmatrix}$$

avec :
[1] : énergie de la structure S_{i+1j+1} où i et j sont les derniers nucléotides appariés avant une boucle terminale.
[2] : énergie minimale de toutes les structures S_{ij} où i et j sont appariés, i' et j' sont appariés, et entre i et i' et j et j' aucun nucléotide n'est apparié. Cela permet de décrire un renflement, une boucle interne ou dans le cas où i'=i+1 et j'=j-1 une zone d'appariement.
[3] : énergie minimale de toutes les structures S_{ij} où i et j sont appariés, qui sont la combinaison de 2 sous structures quelconques $S_{ii'}$ et $S_{i'+1j}$. Cela permet de réaliser des boucles multiples.
w(i,j) est quant à lui calculé ainsi :

$$w(i,j)=min\begin{pmatrix} {}^{1}v(i,j) \\ {}^{2}w(i,j-1) \\ {}^{3}w(i+1,j) \\ {}^{4}min(\forall i', i<i'<j-1, w(i,i')+w(i'+1,j)) \end{pmatrix}$$

avec :
[1] : énergie minimale de toutes les structures possibles de S_{ij} où i et j sont appariés.
[2] : énergie minimale de toutes les structures possibles de S_{ij-1} (décalage à droite).
[3] : énergie minimale de toutes les structures possibles de S_{i+1j} (décalage à gauche).
[4] : énergie minimale de toutes les structures S_{ij}, qui sont la combinaison de 2 sous-structures quelconques $S_{ii'}$ et $S_{i'+1j}$ (boucles multiples).
L'énergie libre minimale est donnée par w(1,n). La (ou les) meilleure(s) structure(s) sont retrouvées en remarquant les parcours possibles de la matrice ayant abouti à la valeur de cette case (stratégie

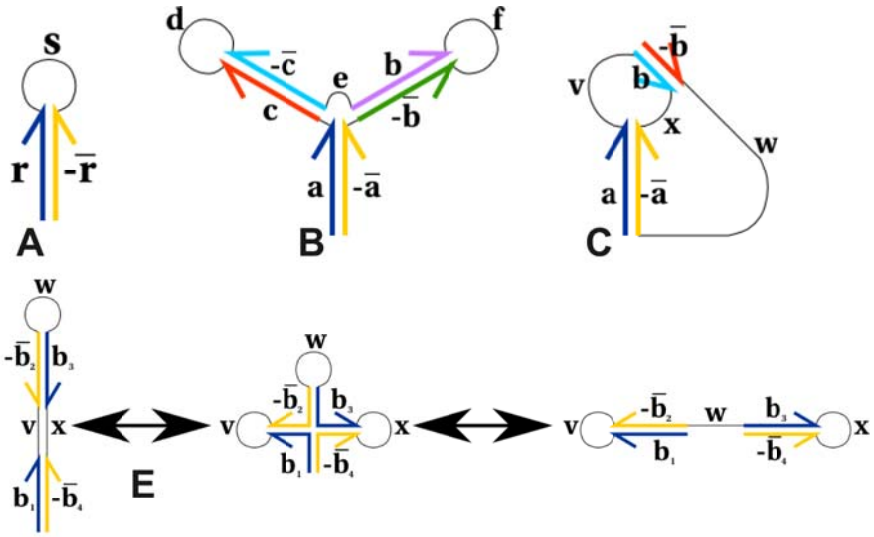


figure 4: Linguistique des Structures Secondaires des séquences d'acides nucléiques

Une structure en tige-boucle (**A**) montre que la grammaire nécessaire pour décrire une structure secondaire de l'ARN est au moins linéaire. Une structure en arborescence (**B**) montre que cette grammaire doit au moins être algébrique. Cependant, il faudra une grammaire contextuelle pour pouvoir décrire un pseudo-nœud (**C**). L'ensemble des structures de l'ARN est descriptible par de telles grammaires. De plus, il est possible d'avoir plusieurs conformations pour une même séquence (**E**),il y a donc des ambiguïtés possibles.

de « back-tracking » ou retour sur trace). C'est un simple parcours de graphe.

2.3.3 Modélisation linguistique des structures secondaires

Les structures précédentes correspondent à un langage particulier sur l'ARN. De manière plus générale, David Searls(15) a étudié quelle expressivité était nécessaire dans le cadre de la théorie des langages pour représenter les structures existantes.

Reprenons l'exemple précédent en introduisant quelques notations. A la base des structures secondaires de l'ARN se situent les séquences répétées inversées complémentaires.

Notons $\sum arn = \{A,C,G,U\}$ l'alphabet de l'ARN, -r la séquence inverse de r et $\bar{r}$ la séquence complémentaire. Soit r = z.v, avec z et v deux séquences d'ARN, z et v $\neq ø$, -r = -(z.v) = -v.-z. $\bar{r} = \overline{(z.v)} = \bar{z}.\bar{v}$.

Dans le cas où l'on considère uniquement les appariements de Watson et Crick (appariement *WC*), $\overline{A}$ = U, $\overline{C}$ = G , $\overline{G}$ = C et $\overline{U}$ = A. Cependant, les appariements Wobble peuvent aussi être pris en compte (i.e. ajout de la liaison G-U), ce qui complexifie un peu les possibilités. On a alors : $\overline{G}$ = (C|U) et $\overline{U}$ = (A|G). D'où, - ACCGC= CGCCA, $\overline{ACCGC}$= UGGCG avec appariement WC et $\overline{ACCGC}$= (UGGCG|UGGUG) avec appariement WC+wobble.

La structure appelée « tige-boucle » est représentée par « r.s.-$\bar{r}$. » (figure 4A). La séquence répétée inversée complémentaire -$\bar{r}$ est capable de s'apparier avec la séquence source pour former une « tige ». s sera la boucle.

Du fait que l'ARN est un objet physique, certaines limites ne peuvent pas être décrites facilement. Les brins ont une « épaisseur », une « souplesse » limitées et le lien entre 2 acides nucléiques a une dimension définie. Ainsi, par exemple le logiciel de calcul de repliement de l'ARN, RNAFold, n'acceptera pas des boucles de taille inférieure à 3 nucléotides. Des séquences peuvent être trop proches ou trop loin pour pouvoir s'apparier correctement ou une torsion peut être trop importante pour être réalisée.

La grammaire pour la tige-boucle est linéaire (elle appartient donc a un sous-ensemble strict des

grammaires algébriques), cependant il est facile d'imaginer des structures plus complexes pour lesquelles ce type de grammaire n'est plus suffisant. En effet, une structure imbriquée comme « a.c.d.-$\bar{c}$.e.b.f.-$\bar{b}$.-$\bar{a}$ » avec a, b, c , d, e et f qui appartiennent à $\sum arn^{+}$, qui décrit une structure qui ressemble à un « tronc » surmonté de 2 « branches » (figure 4B), montre la nécessité de structures algébriques plus générales.
De plus, une séquence peut avoir une analyse ambiguë. En effet, il est possible que plusieurs repliements incompatibles soient réalisables. Par exemple la séquence « b.v.-$\bar{b}$.w.b.x.-$\bar{b}$ », avec b, v, w et x quatre séquences non vides, admet deux analyses de structures alternatives (b.v.-$\bar{b}$)w(b.x.-$\bar{b}$) et (b.v.(-$\bar{b}$.w.b).x.-$\bar{b}$) (figure 4E). En réalité, les structures avec l'énergie la plus faible formeront les structures majoritaires. Du fait que les liaisons G-C sont celles avec l'énergie la plus faible une tige composée d'un fort pourcentage de GC sera très fortement favorisée. D'une façon intéressante, l'ambiguïté est utilisée dans une cellule, ceci permet d'avoir une réponse différente selon les conditions, conditions qui influencent la conformation la plus stable.
La structure appelée « pseudo-nœud » s'exprime par « a.v.b.x.-$\bar{a}$.x.-$\bar{b}$ » (figure 4C). Elle est caractérisée par des séquences répétées inversées complémentaires qui s'entrecroisent (des dépendances croisées). Le fait qu'il puisse y avoir des dépendances croisées impose que l'expression des structures secondaires nécessite une grammaire au moins contextuelle.

2.3.4 Représentation par expressions parenthésées

Les outils de calcul de repliement de l'ARN comme Mfold(17) ou RNAFold(8) ne tiennent pas compte des pseudos-nœuds. Il est très courant de représenter un repliement par l'expression de sa structure algébrique.
Une manière simple de représenter ce type de repliements est d'utiliser un langage de Dyck. On peut se limiter à 3 symboles : '(', ')' et '.' (la parenthèse ouvrante, la parenthèse fermante et le point). Une paire de parenthèses signifie l'appariement des bases à ces positions. Un point signifie que la base à cette position est totalement libre. Une telle représentation est explicite, elle est facile à enregistrer (elle est purement textuelle) et assez simple à utiliser d'un point de vue informatique.
Ainsi l'expression '(((((.(...)))))' désigne une structure en tige boucle dont la tige a une longueur 5, la boucle une longueur 3, et telle que la tige contient un 'renflement' (*bulge*) en position 4.

2.4 Logol, outil de recherche de modèles dans des séquences biologiques

Logol(3) est un outil permettant de chercher des instances de « motifs complexes » (on dira aussi modèles) dans des séquences biologique (ADN, ARN, protéines). En pratique, Logol désigne à la fois un langage de modélisation de motifs biologiques, basé sur une représentation grammaticale, et un outil de recherche des instances du motifs dans les séquences (i.e. un outil de « pattern matching »). Comme vu précédemment, le langage associé aux séquences nucléotidiques permettant d'exprimer des structures secondaires un peu complexes (tel que les pseudo-noeuds) est du type « langage contextuel ».
Le langage Logol est basé sur les « grammaires à variables de chaînes » (String Variable Grammars, SVG), qui est un type de grammaire introduit par David Searls(16) destiné à décrire les séquences d'acides nucléiques à l'aide de variables associées à des sous-chaînes. Le formalisme repose sur les grammaires algébriques, mais l'ajout des variables de chaînes le fait passer dans la catégorie supérieure, celle des langages « faiblement contextuels » (*mildly context-sensitive*).
Le langage Logol définit au plus haut niveau la notion de non terminaux « modèle ». Un « modèle »

est décrit à partir d'autres « modèles » et/ou de « variables ».
Les « variables » représentent des segments de séquences via un ensemble de contraintes : la position de début, la position de fin, la taille, ainsi qu'un intervalle d'acceptation et le contenu. Le contenu peut être lui même défini selon un grand nombre de contraintes : une chaîne de caractère « mère », les substitutions ou plus généralement des « morphismes » qui lui sont applicables. Un morphisme très utilisé est le morphisme « wc » qui permet d'obtenir la chaine complémentaire selon les règles d'appariements de Watson et Crick. Le « - » permet d'obtenir la chaine inverse.
Il est possible de mémoriser des informations concernant une variable dans un « enregistrement ». Ces informations peuvent être utilisées ailleurs, dans une autre variable pour compléter les contraintes permettant de décrire cette dernière.
Plusieurs opérateurs permettent de décrire la manière dont se succèderont les éléments dans un modèle. Il existe deux types d'opérateurs de concaténation, l'opérateur de concaténation classique et l'opérateur de concaténation autorisant un chevauchement avec l'élément précédent. Il y a aussi un opérateur permettant d'exprimer des possibilités multiples (un « ou ») et un opérateur permettant d'exprimer la répétition.
Un tutoriel Logol (http://training.genouest.org/claroline/claroline/learnPath/navigation/viewer.php), portant sur l'éditeur graphique, la grammaire du langage et d'autres documents est accessible en ligne .
La grammaire Logol, ou « **modèle** », peut être donnée sous forme d'une expression de règles (forme grammaticale) ou sous forme graphique.
Cette dernière est réalisable via un éditeur accessible sur le site de la plateforme Genouest (http://webapps.genouest.org/LogolDesigner/javascript/designer/editors/logoleditor.html). Cet éditeur permet de réaliser simplement une grammaire restreinte décrivant le motif recherché. La possibilité de disposer librement chaque élément sur la zone de travail offre la possibilité de dessiner en même temps la structure, rendant très facilement lisible la grammaire.
Enfin, une grammaire en Logol peut être utilisée pour rechercher le motif qu'elle décrit dans un jeu de séquences à l'aide de l'**analyseur Logol**.
Le lancement de l'analyse peut se faire soit à partir de l'interface accessible sur le site de Genouest (http://webapps.genouest.org/org.irisa.genouest.logol.LogolAnalyser/), soit à partir de la version en ligne de commande accessible sur le cluster de Genouest (NDR: méthode plus pratique). L'accès au cluster, se fait via une connexion SSH sur une machine dédiée nommée *le frontal*. Aucun travail requérant de la puissance de calcul ne doit être exécuté sur cette machine. Il faut les exécuter sur les autres nœuds (machines) du cluster à l'aide d'une commande de soumission de travaux « qsub ». Une exécution lancé ainsi est appelée « job ».
L'analyseur prend en entrée le modèle créé par l'une ou l'autre méthode (textuelle ou graphique) et les séquences à confronter au modèle. Les séquences sont dans un fichier au format FASTA, format rappelé ici : chaque séquence est identifiée dans la première ligne (header) et la séquence elle même est écrite sur les lignes suivantes jusqu'au prochain header. Un header doit être écrit sur une seule ligne. Il se distingue par le caractère '>' au début de la ligne. La séquence, quant à elle, est généralement écrite sur plusieurs lignes de taille fixe, le plus souvent 70 ou 80 caractères.
Les résultats produits par l'analyseur Logol sont fournis à l'utilisateur au format XML. Pour chaque séquence, est donnée une description complète du parcours réalisé dans le modèle ayant permis de trouver les matchs.
Des compléments d'informations sur Logol sont présentés en section 5.

3 Recherche et annotation des tiges-boucles dans les repeats de Crisprs

La première étape de la recherche consiste à définir le modèle de tige-boucle adapté à la situation des Crisprs, ce qui est réalisé par affinements successifs. Un premier modèle de tige-boucle, issu de travaux préliminaires, est ainsi affiné suivant différents paramètres jouant un rôle dans la qualité de la tige-boucle : le pourcentage de GC (%GC), la taille de la tige, le nombre de mésappariements et la distance entre les deux parties de la tige (taille de la boucle). La spécificité du modèle est établie par validation statistique sur trois jeux de données: un échantillon représentatif des données à analyser (i.e. des repats de Crisprs), des données générées de façon aléatoire, et des données biologiques non pertinentes (ici les spacers de Crisprs).
Une fois le modèle déterminé, la deuxième étape consiste à mettre en œuvre l'identification des tiges-boucles dans l'ensemble des données cibles. Le modèle est pour l'essentiel exprimé sous forme grammaticale en Logol, et l'analyse syntaxique est effectuée avec l'outil de scan de Logol.
La troisième étape consiste à intégrer les données de structures à Crispi, la base de données de Crispr, sachant qu'une difficulté est d'ajouter ces informations à la base sans trop perturber le schéma existant qui est en exploitation.
Il faut ici déterminer les informations à mettre dans Crispi et celles à afficher à l'utilisateur, ce qui amène à redéfinir en conséquence le schéma relationnel de la base ainsi que le site web.
Il faut également réaliser un programme qui d'une simple commande puisse mettre à jour automatiquement les informations relatives aux tiges-boucles. Ce programme doit s'insérer dans la routine générale de mise à jour de la base.

3.1 Élaboration du modèle des structures secondaires de CRISPR

3.1.1 Constitution des données de validation

Nous avons constitué trois jeux de données pour régler les paramètres du modèle de structures secondaires d'un repeat. Le premier jeu de données est le sous-ensemble des séquences « repeat » de la base CRISPI allant de 25 à 35 nucléotides (33771 séquences), ce qui correspond au modèle standard de repeat de CRISPR. Le second jeu de données est un ensemble de 30000 séquences de 30 nucléotides générées aléatoirement. Enfin, le troisième jeu de données est le sous-ensemble des séquences « spacer » de la base CRISPI allant de 25 à 35 nucléotides (22987 séquences). Ce dernier jeu de données aura pour rôle d'être complémentaire au rôle de témoin négatif des séquences aléatoires. En effet, les séquences des spacers peuvent être assimilées à des séquences biologiques aléatoires en ce qui concerne les structures secondaires(11).
Les repeats et les spacers sont obtenus à partir de la base de données CRISPI. Du fait de la grande fidélité des repeats au sein d'un CRISPR, il a été possible de factoriser les 33771 repeats en 3470 séquences uniques de repeat. Cela permet de réduire par un facteur 10 les temps d'analyse grammaticale pour ce type de séquence. Cela nécessite en contre-partie d'utiliser un fichier permettant de pondérer *a posteriori* les résultats dans les statistiques, mais ce surcoût est assez négligeable face au gain de temps apporté par une telle stratégie. A contrario, le gain est beaucoup plus faible si l'on considère les spacers uniques, ceux-ci étant par définition à peu près uniques (il y a 22109 séquences différentes parmi les 22987 séquences).
Les séquences aléatoires sont générées avec une équiprobabilité pour chaque nucléotide. Cette méthode ne produit cependant pas des séquences ayant des propriétés de séquences biologiques. Il

est en effet très difficile de générer des séquences biologiques, les propriétés de celles-ci étant complexes et pas toujours facile à cerner. Par exemple les séquences biologiques possèdent généralement un taux de GC (%GC) caractéristique pour chaque organisme, et une séquence ayant un %GC différent du génome où il se trouve est le signe d'une séquence d'origine étrangère à l'organisme. Également, grâce à la redondance du code génétique, certains triplets de nucléotides vont être privilégiés pour coder une protéine. Ceci fait que générer aléatoirement des séquences biologiques réalistes n'est pas réellement possible. Les spacers, qui sont les segments intercalés entres les repeats, forment un jeu de séquences naturelles qui ne semblent pas avoir de propriétés spécifiques au niveau de leur structure secondaire. Ils constituent ainsi un bon complément aux séquences aléatoires.

3.1.2 Une première modélisation des tiges-boucles en Logol

Un travail préliminaire réalisé au sein de l'équipe a proposé un premier modèle de tige-boucle (figure 5). Les résultats des repeats confrontés à ce modèle semblaient tout à fait corrects : 96% des séquences réelles étaient reconnues. Cependant, la spécificité n'avait pas encore été testée. La possibilité d'avoir des tiges de seulement quatre nucléotides de long avec au plus un mésappariement (un *mismatch*) laissait présumer une spécificité assez faible. Il faut en effet prendre en compte toutes les possibilités de position de début de la première séquence et la souplesse de la taille de la boucle (de 3 à 8 nucléotides). Afin de vérifier cette spécificité, les séquences aléatoirement générées ont été confrontées à ce modèle. Il est apparu que celui-ci laissait passer 93% de ces séquences. Pire, 95% des spacers matchaient au moins une fois. Partant de cette observation, il est apparu nécessaire d'améliorer le modèle. Dans ce but, un outil d'analyse des résultats Logol a été conçu (cf section 5.3) et plusieurs fois remanié, afin d'analyser la répartition des matchs suivant des angles variés. Parmi les différentes façons de synthétiser les résultats, ont ainsi été évaluées la répartition des matchs en ne prenant que « les meilleures tiges », et la répartition de l'ensemble des matchs en fonction de la taille de la tige, de son pourcentage de GC et de la présence de mésappariements. Des statistiques variées ont ainsi été élaborées pour identifier les caractères spécifiques aux tige-boucles de repeats.
Craignant de produire un modèle trop spécifique des données s'éloignant de la réalité biologique de la tige-boucle, nous avons souhaité pousser l'exploration en étudiant des structures ayant une meilleure garantie d'être biologiquement réalistes. Le problème a donc été abordé via une méthodologie complémentaire, basée sur l'analyse du modèle physique de repliement des structures.

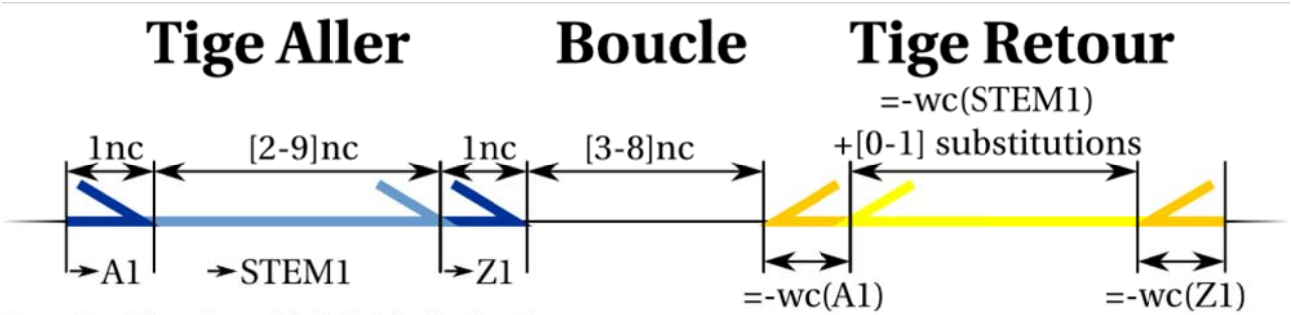


figure 5: schéma du modèle initial de tige-boucle

Ce modèle décrit une tige de 4 à 11 nucléotides avec une substitution possible dans son « corps » avec une boucle de 3 à 8 nucléotides. Il permet d'avoir la certitude que les extrémités soient bien complémentaires. Grammaire Logol associée:

```
Mod1() ==> A1:{#[1,1],_LA1}, STEM1:{#OPT[2,9],_LSTEM1}, Z1:{#[1,1],_LZ1}, LOOP1:{#[3,8]},
-"wc" ?LZ1,  -"wc" ?LSTEM1:{$[0,1]}, -"wc" ?LA1
```

3.1.3 Modélisation à partir de données RNAFold

Une méthode usuelle de prédiction de structures secondaires de l'ARN est basée sur le calcul de l'énergie libre associée aux repliements, les repliements les plus probables étant ceux de plus faibles énergie. Il peut exister plusieurs solutions exhibant une énergie libre proche de l'énergie minimale.
Nous avons effectué les simulations de repliement des structures présentes sur nos séquences à l'aide du logiciel RNAFold , un des logiciels spécialisés sur ce type de calcul. Ce logiciel utilise le modèle de Zuker *et al* (17) pour calculer les repliements. Les repliements obtenus sont le résultat d'une approximation du modèle de calcul de l'énergie libre du repliement de l'ARN sur lui-même. Il n'est proposé qu'un seul résultat, le repliement avec la plus faible énergie libre, qui est le repliement majoritaire. Les autres repliements existent tout de même. RNAFold peut donc parfois retourner des structures trop faibles. Il est espéré que ces approximations soient lissées par le grand nombre de séquences.
Il existe d'autres logiciels capables de réaliser le même travail, comme Mfold par exemple. Mfold et RNAFold sont tous les deux déployés sur le cluster de Biogenouest. Les deux outils réalisent exactement les même résultats dans le cadre de l'étude actuelle. Le choix s'est porté sur RNAFold car ce logiciel a été utilisé lors d'une étude précédente sur le même sujet(11).
Les jeux de données sont soumis au logiciel RNAFold à l'aide d'un script bash exécuté sur le cluster. Parmi les options possibles, les appariements G-U (appariement Wobble) ont été autorisés. Pour chaque séquence, RNAFold retourne trois fichiers : un premier fichier texte au format Vienna, et deux fichiers Postscript donnant deux autres représentations du repliement (une matrice avec des repliements alternatifs et une version visuelle). Le format Vienna propose une description du repliement à l'aide d'une expression bien parenthésée. Les tiges-boucles seront extraites de cette dernière. Pour se faire, nous avons conçu un analyseur spécifique.
Une tige-boucle est définie ici par une succession de paires de nucléotides sans mésappariements surmontée d'une boucle terminale. Cela se traduit dans une expression bien parenthésée par une succession sans interruption de parenthèses ouvrantes suivies d'une série de points puis suivie d'une suite de parenthèses fermantes en nombre égal au nombre de parenthèses ouvrantes :

$$TigeBoucle \rightarrow (^{n}(.^{x})^{n}$$

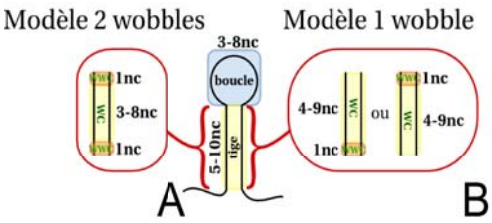
L'analyse d'expressions parenthésées nécessite en temps normal un automate LR. Cependant, étant donnée la simplicité des expressions, nous avons préféré abandonner l'analyse gauche droite pour partir directement des parenthèses les plus emboitées. L'idée est de chercher une succession de points entourée d'au moins une parenthèse ouvrante et une parenthèse fermante. Une fois un tel motif trouvé, il suffit de « consommer » de manière symétrique uniquement les parenthèses ouvrantes à gauche, fermantes à droite et de s'arrêter dès que ceci n'est plus possible. Il est possible ainsi de retrouver les positions de début et de fin des différentes parties de la tige-boucle. Avec ces informations et la séquence complète, il est possible de retrouver les séquences de chaque partie, en particulier celle des deux parties de la tige[4].
Les informations ainsi extraites sont synthétisées dans différents tableaux exprimant la taille des tiges en fonction du pourcentage de GC, la répartition de la taille des boucles, la répartition du nombre d'appariements wobble ainsi que leur position sur la tige.
Le fait que les séquences aléatoirement générées possèdent pour certaines des tiges-boucles tout à fait convaincantes est statistiquement normal, et ces évènements sont considérés comme du bruit de fond. Le but est de réduire le plus possible ce bruit et de conserver le maximum de résultats positifs dans le cas des repeats.
Il a été ajouté un système de filtres sur différents critères pour pouvoir pré-tester des modèles et

4 Cet outil a été implémenté en Java en reprenant une partie de l'analyseur de résultats Logol (cf page 23)

vérifier que ceux-ci éliminent le maximum des témoins négatifs sans pour autant trop réduire le nombre de repeats valides.
En partant des observations et en comparant les résultats des repeats vis-à-vis des spacers et des séquences aléatoires, les tiges de taille inférieure à quatre nucléotides ont été éliminées, ce critère étant majoritairement présent dans les témoins négatifs. Après différents essais, le taux de GC dans la tige (la concaténation des deux séquences composant la tige) est apparu être autour de 60%. Enfin, il a été remarqué que le nombre de wobbles est généralement inférieur ou égale à deux et que ce wobble se trouve généralement aux extrémités.
Un modèle Logol est alors déterminé grâce à ces informations. Les résultats obtenus pour les séquences aléatoires et les séquences des spacers (les témoins négatifs) permettent de repérer les éléments du modèle sur lesquelles jouer pour améliorer encore la spécificité de ce modèle (figure 6A).
Comme attendu, il est observé une similarité des résultats entre les séquences aléatoirement générées et les séquences des spacers. Cependant les séquences des spacers se distinguent légèrement des séquences aléatoires par une proportion de tige-boucles légèrement plus importante et de meilleure qualité. Il est assez difficile de tirer des conclusions d'une telle observation tant l'enchaînement des nucléotides dans une séquence biologique est complexe et soumise à un grand nombre de facteurs.

3.1.4 Modèle Logol final pour les structures secondaires

Deux modèles Logol ont finalement été réalisés (figure 6A), l'un autorisant jusqu'à deux wobbles, un à chaque extrémité, l'autre un seul wobble, à l'une seulement des extrémités. Aucun autre type de mésappariement n'a été autorisé.
Du fait que l'analyseur Logol retourne plusieurs matchs, il a été choisi que les meilleures tiges-boucles seraient sélectionnées dans l'ordre de priorité décroissante suivant : les tiges les plus grandes, les tiges avec le moins de mésappariements et enfin, la boucle la plus petite. Si ces critères ne permettent pas de départager plusieurs matchs, alors un des matchs sera pris au hasard. Le taux de GC étant déjà dans les critères et le taux espéré étant déjà très élevé, ce critère a été jugé mineur.
Les seuls mésappariements autorisés par rapport aux appariements de Watson et Crick sont les wobbles. Ces « mésappariements » se traduisent par l'ajout d'une énergie négative dans le compte de l'énergie libre minimale. La taille de la tige est donc un critère plus important que le nombre de wobbles. La taille de la boucle est un critère assez arbitraire, elle repose sur le fait que plus les séquences sont éloignées, moins elles ont de chance de s'apparier.



	Modèle 1 Wobble			Modèle 2 Wobbles		
%GC	Repeat	Spacer	Aléatoire	Repeat	Spacer	Aléatoire
0%	55,0%	89,5%	93,8%	46,5%	79,5%	84,2%
50%	57,6%	93,3%	95,9%	52,6%	85,0%	90,3%
60%	57,7%	94,9%	97,7%	52,6%	87,7%	93,1%
70%	58,4%	95,5%	98,3%	59,4%	90,9%	96,1%

figure 6: Résumé des différents modèles testés

Deux modèles sont candidats (**A**). Ils décrivent tous les deux une tiges-boucle dont la tige fait 5 à 10 nucléotides de long et une boucle de 3 à 8 nucléotides. La tige est composée de 2 brins complémentaires. Les wobbles ne peuvent être présents qu'aux extrémités de la tige. Un des modèles autorise au plus 1 wobble à chaque extrémité (Modèle 2 wobbles), l'autre 1 seul wobble à l'une ou l'autre des extrémités (Modèle 1 wobble). Pour chacun de ces modèles, différents pourcentages de GC ont été testés (**B**), le pourcentage de séquences ne répondant pas aux critères est alors comptabilisé. Le modèle 1 wobble avec 60% de GC dans la tige est celui qui est apparu le meilleur.

Afin d'obtenir le meilleur rapport entre perte de séquences possédant au moins un match et élimination du bruit de fond, des filtres de 0% (sans filtre), 50%, 60% et 70% de GC pour la tige ont été appliqués aux résultats des deux modèles (figure 6B) (cf la remarque à ce sujet en section 5.3).
Au final, le modèle proposant **un seul wobble et un taux de GC de 60% et plus** s'est montré suffisamment spécifique et éliminant pratiquement 94.9% des spacers et 97,7% des séquences aléatoires mais en conservant 42,4% des repeats. Un seuil d'erreur acceptable de 5% est un seuil couramment utilisé. Avec le critère choisi, le modèle s'approche de ce seuil de spécificité. Le modèle Logol final est présenté en figure 11.
Dans la base de données, chaque CRISPR possède une séquence consensus. Cette séquence est sensée représenter une séquence « moyenne » de tous les repeats du CRISPR. Du fait que les repeats sont très fortement conservés au sein d'un CRISPR, les séquences consensus sont pour la plupart très similaires aux repeats. De ce fait, il est attendu que la structure de la tige boucle du consensus et celles des repeats associés soit similaires. Pour pouvoir vérifier cette hypothèse, nous avons extrait les séquences consensus de la base de données ainsi qu'un fichier XML permettant de relier l'identifiant d'une séquence consensus à l'ensemble des identifiants de ses séquences repeats. L'affichage de la séquence et de la structure au-dessus de celles de ses repeats a permis de vérifier cette hypothèse. Il est même apparu que la position centrale des structures vis-à-vis des séquences semblait toujours la même dans un CRISPR (séquence consensus et repeats). Une autre observation qui demanderait confirmation a été faite, les différences entre les séquences, lorsque celles-ci portaient sur les tiges-boucles, se retrouvaient *a priori* au niveau de la boucle, i.e. les tiges semblent généralement inchangées.

3.2 Modification du schéma de la base de données CRISPR

Le modèle de tige-boucle dédié aux repeats de Crisprs étant établi, il s'agit maintenant de mettre en œuvre l'annotation de ces structures dans la base CRISPI.

3.2.1 Schéma d'origine

La base de données CRISPI est une base de données relationnelles réalisée avec le SGBD MySQL. Son schéma est présenté en figure 7 (partie en noir). On peut remarquer que le schéma contient certaines redondances d'informations, en effet lors de l'établissement de ce schéma relationnel, il a été décidé de factoriser les séquences uniques des spacers et de repeats dans une même table. Cette factorisation permet de réduire la redondance d'information, en particulier des repeats. Le fait que les spacers et les repeats soient dans une même table permet de retrouver dans l'ordre les différents éléments d'un seul CRISPR à l'aide d'une requête simple.
Les données cibles, dans lesquelles seront recherchées les tige-boucles, sont l'ensemble des repeats du « train de repeats » d'un Crispr, ainsi que la séquence consensus associée. Les repeats sont localisés dans la table DATA; ils ont le type 'repeat'. Les consensus quant-à-eux sont localisés dans la table CRISPR (CRISPR.consensus_seq).
Une contrainte particulière dans l'insertion des tiges-boucles dans la base de données est de ne pas « démonter» l'existant, la base étant utilisée en exploitation. De plus, les données étant destinées à être affichées sur un site web, il est nécessaire de garantir une certaine réactivité et donc de ne pas complexifier inutilement leur accès. Une autre contrainte venant du schéma lui-même est de réduire au maximum les « cases vides ». Au total, il faut choisir le meilleur compromis possible.

3.2.2 Solutions d'évolution possibles

Les informations utiles sur les tiges boucles à ajouter dans la base de données sont :

- la position de début par rapport à la séquence « hôte »;
- la structure représentée à l'aide d'une expression parenthésée (permet l'évolution du modèle en acceptant des tige-boucles imparfaites);
- la taille de la tige : cette information pourrait être extraite de l'expression parenthésée, mais pas en SQL;
- la taille de la boucle : de même que pour la taille de la tige;
- la séquence de la tige-boucle : cette information pourrait être extraite des informations précédentes et de la séquence associée (DATA.sequence ou CRISPR.consensus_seq), mais pas en SQL.

Au vu des contraintes, pour pouvoir intégrer ces informations, plusieurs solutions ont été imaginées .

La première consiste à ne pas créer de table, mais à ajouter les informations relatives aux tige-boucles au sein des tables déjà existantes. Ainsi les tables DATA pour les repeats et CRISPR pour les séquences consensus seraient modifiées pour accueillir l'intégralité des informations concernant les tiges-boucles associées à leurs séquences.

La seconde possibilité consiste à créer une table StemLoop. Cette solution propose de mettre l'accent sur la structure de la tige-boucle en plaçant les informations relatives à la structure secondaire et à la séquence de la tige-boucle dans une table séparée de la séquence associée (DATA.sequence ou CRISPR.consensus_seq). Cela nécessite quand même de modifier les tables DATA et CRISPR pour ajouter une clé étrangère vers une entrée de cette table et la position de début par rapport à la séquence « hôte ».

La troisième possibilité proposée reprend la table StemLoop de la seconde mais cette fois-ci les relations sont réalisées à l'aide de deux tables associatives. Cette solution contrairement aux deux

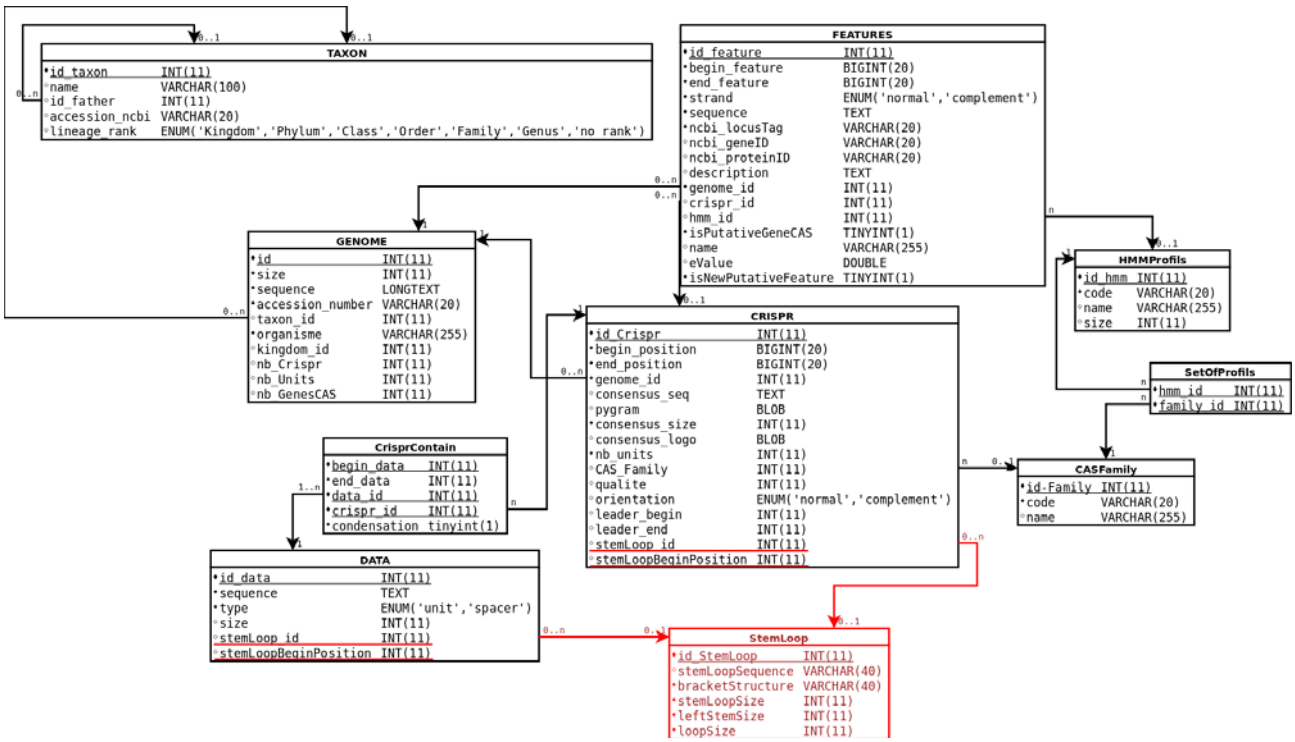


figure 7: Nouveau schéma de la base de données CRISPI

Les modifications apportées pour pouvoir intégrer les informations relatives aux tiges-boucles sont représentées en rouge.

autres évite de modifier les tables DATA et CRISPR.
La première proposition a été refusée du fait qu'elle provoquerait une grande quantité de redondances. En effet, considérant la tige-boucle comme une entité identifiable à l'aide de sa structure et de la séquence associée, ces informations, ainsi que les informations associées, seraient dupliquées. De plus, cette proposition aurait pour effet d'ajouter beaucoup de colonnes dans les tables, colonnes qui seraient bien souvent vides du fait de la proportion des séquences repeats à avoir des tige-boucles, et du fait que les entrées dans la table DATA du type repeat représentent à peine 10% de la table.
La seconde proposition a l'avantage d'éviter les redondances et de mettre en valeur la tige-boucle elle-même en dehors du contexte mais ne permet pas de supprimer toutes les colonnes inutiles. En effet, il est toujours nécessaire de placer une clé étrangère et la position de début de la tige boucle dans les tables DATA et CRISPR.
La troisième proposition règle ce dernier problème mais augmente grandement la complexité de la base de données. Cette augmentation de la complexité est injustifiée pour réaliser une relation un à plusieurs (0..1:0..n) alors qu'une table associative est là pour réaliser des relations plusieurs à plusieurs.
La seconde solution, qui ne déstructure pas trop la base existante, a finalement été jugéle meilleur compromis. Nous avons considéré que l'incohérence de la présence des colonnes stemLoop_id et stemLoopBeginPosition dans la table DATA dans les entrées de type « spacer » n'était pas si gênante. Il suffit de ne pas les considérer dans ce cas. Toujours dans ce cas, les scripts modifiant la base de données doivent tout de même assurer que ces champs restent à null.

3.2.3 Nouveau schéma de la base

La solution retenue pour le nouveau schéma de la base consiste donc à placer les informations relatives à la structure secondaire et à la séquence de la tige-boucle dans une table séparée de la séquence associée (i.e. de la séquence repeat); le schéma résultant est visible en figure 7. Une table StemLoop est ainsi créée, qui contient les caractéristiques des tige-boucles rencontrées. A un identifiant de tige-boucle y sont associés: la chaîne de nucléotides concernée, la structure sous forme parenthésée, la taille globale de la tige-boucle, la taille de la partie gauche de la tige (qui pourrait être différente de la partie droite en cas d'indel dans la tige) et la taille de la boucle.
Les séquences auxquelles se rapportent ces structures, c'est-à-dire les séquences de repeats, sont inventoriées dans deux tables, DATA et CRISPR: les repeats des Crispr sont enregistrés de la table DATA (DATA.sequence pour les enregistrements de type 'repeat'), et les consensus des repeats sont enregistrés dans la table CRISPR (CRISPR.consensus_seq). Deux attributs sont donc ajoutés à ces tables: l'identifiant de la structure de tige-boucle présente dans la séquence (pointeur vers la table Stemloop) et la position de début par rapport à la séquence.

3.3 Mise à jour automatique de la base de données

3.3.1 Mise à jour intégrée dans une routine de mise à jour générale

La base de données Crispi est mise à jour tous les mois. Cette mise à jour est réalisée à l'aide du logiciel BioMAJ (http://biomaj.genouest.org/). Une liste des génomes de la base de données NCBI qui ont soit été modifiés, soit ajoutés depuis la dernière mise à jour, est établie. Seuls les génomes complets d'archées ou de bactéries sont conservés. Ces génomes sont téléchargés sur le cluster.

Parmi les fichiers téléchargés, il y a un fichier avec l'extension « .fna » correspondant au fichier FASTA où se trouve la séquence complète du génome. Un fichier texte « diff.txt » contenant le chemin vers tous les fichiers « .fna » des génomes à mettre à jour est créé.
Le script de mise à jour des tiges-boucles dans la base de données s'intercale dans cette routine. On considère que les CRISPR (table CRISPR et table DATA) sont déjà à jour lorsqu'il est exécuté. Le script doit pouvoir réaliser toutes ces tâches de manière autonome. Le seul point de départ est le fichier texte « diff.txt » créé lors de la mise à jour.
Il faut aussi pouvoir faire une mise à jour complète. Cette fonctionnalité est non seulement nécessaire lors du déploiement de l'outil afin de remplir les tables pour la première fois avec l'intégralité de la base de données, mais cette option peut être aussi utile lorsqu'un des outils de la routine, en amont, est modifié au point de changer ses résultats. Il faut alors refaire l'analyse sur l'intégralité de la base de données.

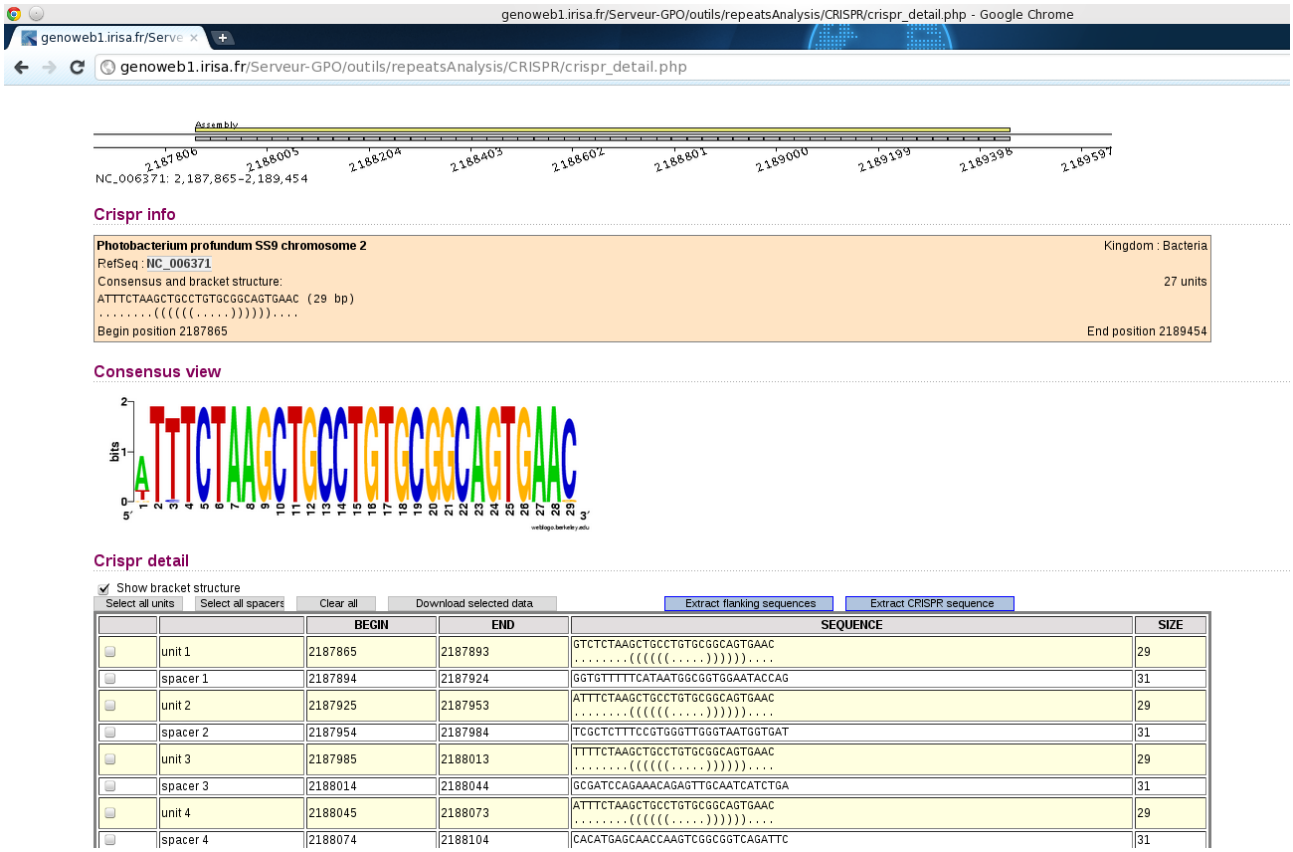
3.3.2 Étapes du traitement

Le programme utilise l'analyseur Logol et le modèle de tige-boucle qui a été choisi précédemment (modèle « 1 wobble et tige à 60% de GC »). Il doit être capable de réaliser une succession de tâches dans un ordre précis.
Tout d'abord, il faut retrouver, à partir des informations contenues dans le fichier « diff.txt », les séquences à mettre à jour et créer des fichiers FASTA pour pouvoir les donner à l'analyseur Logol. La première difficulté est de retrouver ces séquences avec uniquement ce fichier. En fait, le nom des fichiers des génomes correspond à son numéro d'accession NCBI. A partir de ce numéro, il est donc possible de retrouver le génome et les séquences. Un autre problème subsiste : comment à la fin de la mise à jour identifier quelle entrée est associée à la réponse Logol? Dans le fichier XML de résultats Logol, seul l'entête de la séquence est conservé quelque soit le résultat. Ainsi, pour savoir à quelle table, DATA ou CRISPR, appartient la séquence, deux fichiers FASTA sont créés contenant respectivement les séquences provenant de la table DATA et les séquences de la table CRISPR. Pour retrouver l'entrée, il suffit de mettre en entête de chaque séquence la clé de l'entrée qui lui est associée.
Ensuite les fichiers FASTA ainsi créés sont confrontés au modèle avec l'aide de l'analyseur Logol.
Une fois les exécutions terminées, les deux séries de résultats sont analysées, les données sont converties pour ne retenir que les informations à mettre à jour et finalement la base de données est mise à jour. Enfin, il est nécessaire de nettoyer les entrées de la table StemLoop qui ne sont plus référencées ni dans la table CRISPR ni dans la table DATA.
Des détails sur la mise en œuvre des procédures de mise à jour sont données en section 5.4.

3.4 Modification du site web

3.4.1 Environnement technique du site existant

Le site web de la base Crispi est codé en PHP/HTML/javascript, avec l'utilisation de la bibliothèque javascript jQuery. L'architecture de ce site est assez simple. Elle consiste à coupler une adresse à un fichier de script PHP qui génère la page en HTML et javascript. Ces scripts utilisent quelques inclusions pour intégrer d'autres scripts. Parmi les fichiers inclus, le fichier « DB_requetes.php » réunit la plupart des requêtes vers la base de données ainsi que la conception de quelques éléments graphiques (des tables par exemple) qui nécessitent les retours des requêtes pour être construits. Cette façon de faire permet, lorsque le site est de petite taille, de ne pas s'encombrer d'architectures lourdes à réaliser rendant son développement rapide et l'ajout de nouvelles fonctionnalités simples.



		BEGIN	END	SEQUENCE	SIZE
☐	unit 1	2187865	2187893	GTCTCTAAGCTGCCTGTGCGGCAGTGAAC .....((((((.....))))))....	29
☐	spacer 1	2187894	2187924	GGTGTTTTTTCATAATGGCGGTGGAATACCAG	31
☐	unit 2	2187925	2187953	ATTTCTAAGCTGCCTGTGCGGCAGTGAAC .....((((((.....))))))....	29
☐	spacer 2	2187954	2187984	TCGCTCTTTCCGTGGGTTGGGTAATGGTGAT	31
☐	unit 3	2187985	2188013	TTTTCTAAGCTGCCTGTGCGGCAGTGAAC .....((((((.....))))))....	29
☐	spacer 3	2188014	2188044	GCGATCCAGAAACAGAGTTGCAATCATCTGA	31
☐	unit 4	2188045	2188073	ATTTCTAAGCTGCCTGTGCGGCAGTGAAC .....((((((.....))))))....	29
☐	spacer 4	2188074	2188104	CACATGAGCAACCAAGTCGGCGGTCAGATTC	31

figure 8: page de détail d'un CRISPR après modification

Les structures secondaires sont représentées sous la forme d'une expression bien parenthésée en dessous de chaque séquence. Il est possible de choisir d'afficher ou pas les structures grâce à une case à cocher et un code Javascript. Cette case permet aussi de choisir si les structures seront importées ou pas lors de l'extraction des séquences dans un fichier.

3.4.2 Modifications du site

Les modifications réalisées sur le site (cf figure 8) peuvent être divisées en deux catégories, les modifications visuelles du site web et les modifications dans la génération des fichiers d'importation de données. Dans les deux cas, elles consistent à mettre en-dessous d'une séquence sa structure secondaire sous forme d'expression parenthésée. Pour conserver une grande lisibilité sur le site et la compatibilité avec le format FASTA, il est nécessaire de pouvoir désactiver l'affichage des structures secondaires à l'aide de boutons. Sur le site web, cette fonctionnalité ne nécessite pas de recharger la page, un simple script Javascript se charge de faire apparaître ou disparaître les expressions parenthésées.

Afin que les séquences et les expressions parenthésées soient bien alignées, il est nécessaire d'utiliser une police du type « monospace » où chaque caractère occupe la même largeur. Il s'avère utile de notifier explicitement l'absence de tige-boucle. A cet effet, ont été ajouté un message sur le site web et une simple série de points (équivalent de l'absence de structure) dans le fichier importé.

4 Conclusion et perspectives

Affiner un modèle biologique est un processus délicat, qui demande l'intégration de compétences mathématiques et biologiques. Ainsi, la définition d'un modèle caractéristique des tige-boucles de nos séquences cibles a constitué la partie sensible de notre étude. La seule utilisation des connaissances sur les repliements de l'ARN ne s'est pas avérée suffisante pour établir un modèle crédible des structures en tige-boucles présentes dans les Crisprs. Pour aboutir à un modèle satisfaisant (biologiquement plus crédible), il nous a fallu utiliser en complément un logiciel de repliement intégrant un calcul d'énergie libre, RNAFold.

Le modèle retenu permet de détecter une tige-boucle dans 42,4% de nos séquences cibles, les repeats de Crispr. Ne connaissant pas le taux de repeats de Crisprs formant réellement une tige-boucle, la sensibilité de ce modèle ne peut pas être calculée. Cependant, la spécificité du modèle, avec pratiquement 95% d'élimination des matchs dans les séquences négatives, semble garantie.

Certains détails du modèle obtenu peuvent peut-être encore être affinés. En effet, ce modèle autorise au plus un appariement wobble dans la tige, situé à une des extrémités de la tige. En pratique, on observe que la majeure partie des wobbles trouvés sont localisés à la base de la tige. On peut se demander s'il s'agirait d'un biais introduit par l'utilisation faite de RNAfold (pour rappel, l'approche employée lors de la mise au point du modèle consiste à « découper » la tige-boucle inscrite dans la structure plus complexe retournée par RNAfold). Ainsi, l'appariement wobble pourrait être dû à une simple boucle interne à la base de cette tige, c'est-à-dire un espace entre deux appariements. Une autre interprétation envisageable est que ce wooble permette d'assouplir la taille de la tige ou de la boucle et que ce puisse être utile dans le mécanisme d'interaction avec une autre molécule.

D'un point de vue technique, ces travaux ont permis de confronter avec succès notre outil de pattern matching Logol à un véritable problème biologique, tout en permettant de faire évoluer l'outil sur plusieurs points. Cet aspect constituait un des objectifs scientifiques de notre étude.

La base de données CRISPI a été modifiée. Le script de mise à jour des tiges-boucles est déployé et s'intercale dans la routine de mise à jour générale de la base. L'annotation automatique dans CRISPI des tige-boucles de repeat contenu dans l'ensemble des génomes procaryotes est donc pleinement opérationnelle. Cela contribue à renforcer le positionnement de la base CRISPI comme un outil essentiel pour la recherche sur les Crisprs.

Concernant les prochaines modifications envisagées sur la base CRISPI, il est prévu notamment d'ajouter des annotations concernant l'origine des spacers. En effet, bien qu'il soit connu que les spacers sont pour la plupart d'origine exogène, certains seraient la copie d'une portion de génome hôte. Cela laisse à supposer que les CRISPR pourraient avoir un rôle dans la régulation de gènes et provoquer une certaine diversité dans une population phylogénétiquement proche. Un tel type de spacer pourrait faire partie de l'arsenal des mutations rapides que les bactéries utilisent pour s'adapter rapidement à un milieu. Il est ainsi prévu de rechercher les spacers qui auraient une séquence similaire dans le génome pour pouvoir dans un second temps repérer les gènes à proximité de cette séquence. Une ébauche de travail a déjà été réalisée dans cette optique en confrontant, à l'aide de l'outil BLASTn de la suite BLAST, un génome à ses spacers et un des génomes d'*Escherichia Coli* K12 à une série de séquences aléatoirement générées.

De façon plus générale, de nombreuses informations concernant les CRISPRS restent à extraire des séquences, et sont de nature assez diverse, en voici une liste non exhaustive : recherche de zone dites « leader » en amont des séquences, recherche de CRISPR de petite taille difficilement détectables, recherche de correspondances dans les banques publiques de génomes de virus, typologie des CRISPR basée sur les profils de gènes associés (gènes CAS), indice de qualité des prédictions effectuées.

References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. Basic local alignment search tool. J. Mol. Biol. 215, 403-10(1990).
2. Barrangou, R., Fremaux, C., Deveau, H., Richards, M., Boyaval, P., Moineau, S. et al. CRISPR provides acquired resistance against viruses in prokaryotes. Science 315, 1709-12(2007).
3. Belleannée, C. & Nicolas, J. Logol : Modelling evolving sequence families through a dedicated constrained string language. , 19(2007).
4. Deveau, H., Barrangou, R., Garneau, J.E., Labonté, J., Fremaux, C., Boyaval, P. et al. Phage response to CRISPR-encoded resistance in Streptococcus thermophilus. J. Bacteriol. 190, 1390-400(2008).
5. Filangi, O., Beausse, Y., Assi, A., Legrand, L., Larre, J., Martin, V. et al. BioMAJ: a flexible framework for databanks synchronization and processing. Bioinformatics 24, 1823-1825(2008).
6. Grissa, I., Vergnaud, G. & Pourcel, C. CRISPRFinder: a web tool to identify clustered regularly interspaced short palindromic repeats. Nucleic Acids Res. 35, W52-7(2007).
7. Grissa, I., Vergnaud, G. & Pourcel, C. The CRISPRdb database and tools to display CRISPRs and to generate dictionaries of spacers and repeats. BMC Bioinformatics 8, 172(2007).
8. Gruber, A.R., Lorenz, R., Bernhart, S.H., Neuböck, R. & Hofacker, I.L. The Vienna RNA Websuite. Nucleic Acids Res. 36, W70-W74(2008).
9. Horvath, P. & Barrangou, R. CRISPR/Cas, the immune system of bacteria and archaea. Science 327, 167-70(2010).
10. Jansen, R., Embden, J.D.A.V., Gaastra, W. & Schouls, L.M. Identification of genes that are associated with DNA repeats in prokaryotes. Mol. Microbiol. 43, 1565-75(2002).
11. Kunin, V., Sorek, R. & Hugenholtz, P. Evolutionary conservation of sequence and secondary structures in CRISPR repeats. Genome Biol. 8, R61(2007).
12. Mojica, F.J., Ferrer, C., Juez, G. & Rodríguez-Valera, F. Long stretches of short tandem repeats are present in the largest replicons of the Archaea Haloferax mediterranei and Haloferax volcanii and could be involved in replicon partitioning. Mol. Microbiol. 17, 85-93(1995).
13. Nicolas, J.,Durand, P., Ranchy, G., Tempel, S., Valin, A.S. Suffix-tree analyser (STAN): looking for nucleotidic and peptidic patterns in chromosomes. Bioinformatics 21, 4408-4410(2005).
14. Rousseau, C., Gonnet, M., Le Romancer, M. & Nicolas, J. CRISPI: a CRISPR interactive database. Bioinformatics 25, 3317-8(2009).
15 Searls, D.B. in () pp. 47-120 (American Association for Artificial Intelligence, 1993).
16. Searls, D.B. String variable grammar: A logic grammar formalism for the biological language of DNA. The Journal of Logic Programming 24, 73 - 102(1995).
17. Zuker, M. & Stiegler, P. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Res. 9, 133-48(1981).

Crédits images

http://fr.wikipedia.org/wiki/Fichier:Human-gender-neutral.png,
http://fr.wikipedia.org/wiki/Fichier:Biological_cell.svg (auteurs MesserWoland et Szczepan1990),
http://fr.wikipedia.org/wiki/Fichier:Diagramme_d_une_cellule_procaryote_fr.svg.
p5 figure 3 : d'après les images http://fr.wikipedia.org/wiki/Fichier:TRNA-Phe_yeast_1ehz.png (auteur Yikrazuul) et http://en.wikipedia.org/wiki/File:TRNA-Phe_yeast_en.svg (auteur Yikrazuul)

5 Annexe technique

5.1 Exemple de grammaire Logol

Logol permet de chercher des instances de motifs complexes (on dira aussi modèles) dans des séquences biologique (ADN, ARN, protéines). Un modèle Logol pourra être décrit sous forme textuelle (par des règles de grammaire) ou sous forme graphique (cf figure 9) qui sera convertie automatiquement en forme textuelle.
Voici, à titre d'illustration, un exemple de **modèle Logol sous forme textuelle**.

```
model1()==*>SEQ1
model1()==>model2(),model3()
model2()==>« aaa »:{$[0,1]}
model3()==>« acgt »
```

`« aaa »:{$[0,1]}` signifie : rechercher un motif correspondant à un motif « aaa » où il peut y avoir entre 0 et 1 substitution. $ est le symbole de la substitution, il est suivi par un intervalle de substitution accepté. Les segments tels que « aag »,« aca »,« taa » seront donc reconnus ici, mais pas « tta » par exemple. Dans un modèle, la virgule symbolise la succession directe. Dans cet exemple, le motif recherché est découpé en 2 sous-modèles. Le découpage hiérarchique en sous-modèles permet ainsi d'effectuer une description structurée des motifs complexes; les parenthèses permettent d'effectuer des passages de paramètres -non illustré ici. Le motif décrit par model1 peut se paraphraser : 'Recherche du motif « aaa » - avec une substitution possible- suivi de « acgt »'. Parmi les instances de model1 se trouvent notamment « aagacgt » et« acaacgt ». En pratique, le lancement de l'analyse d'une séquence par cette grammaire indiquera l'emplacement de toutes instances de model1 dans la séquence. Ainsi, l'analyse de la séquence « tcaagacgtattgacaacgta » produira deux solutions: « tc**aagacgt**attgacaacgta » et « tcaagacgtattg**acaacgt**a ».

Voici en bleu un modèle grammatical plus direct du même motif, ainsi que la version équivalente sous forme de **modèle Logol graphique** (cf figures 9 et 10).

```
Model1()==*>SEQ1
model1()==>« aaa »:{$[0,1]},« acgt »
```

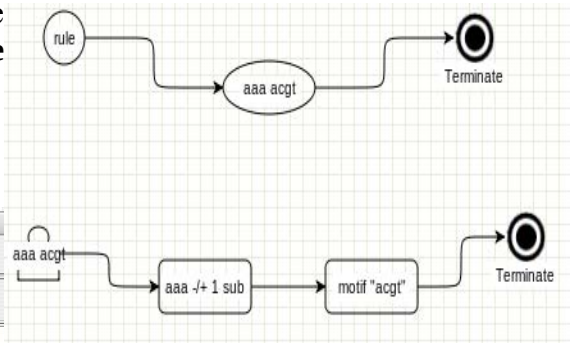


figure 9: Modèle Logol graphique

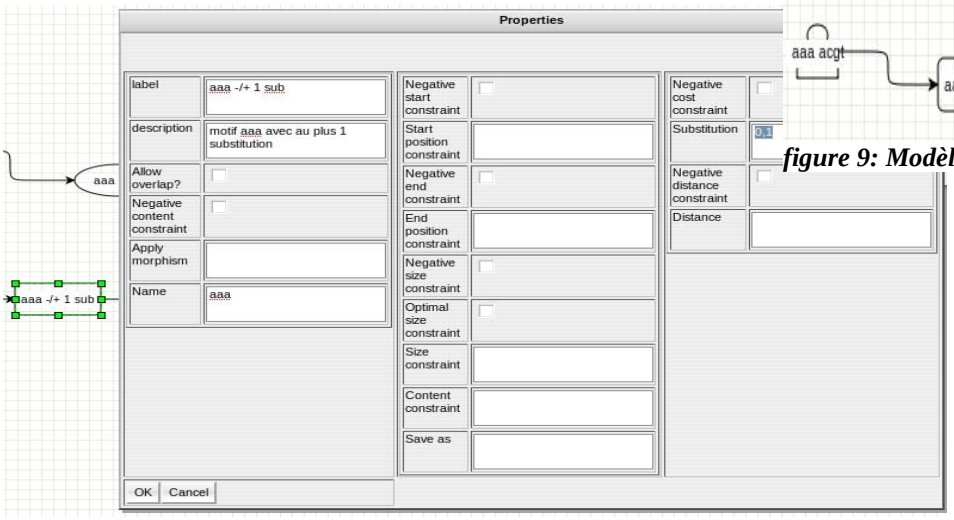


figure 10: Modèle graphique : détail de la variable 'AAA avec au plus 1 substitution'

```
def:{
morphism(wobble,a,t)
morphism(wobble,a,u)
morphism(wobble,t,a)
morphism(wobble,u,a)
morphism(wobble,c,g)
morphism(wobble,g,c)
morphism(wobble,g,t)
morphism(wobble,g,u)
morphism(wobble,u,g)
morphism(wobble,t,g)
}
controls:{
% "cg"[mod1.LSTEM1,mod1.RSTEM1,mod1.LW1,mod1.RW1]>=60
}
mod1()==>((W1:{#[1,1],_LW1},STEM1:{#OPT[4,9],_LSTEM1},LOOP1:
{#[3,8]},-"wc" ?LSTEM1:{_RSTEM1},-"wobble" ?LW1:{_RW1})|(STEM1:
{#OPT[4,9],_LSTEM1},W1:{#[1,1],_LW1},LOOP1:{#[3,8]},-"wobble" ?LW1:
{_RW1},-"wc" ?LSTEM1:{_RSTEM1}))
mod1()==*>SEQ1
```

figure 11: Modèle Logol final : tige-boucles de CRISPR

5.2 Compléments d'informations sur l'utilisation de Logol

L'utilisation du langage Logol pour la modélisation des tige-boucles des Crisprs a permis de pointer de nouveaux besoins par rapport à l'outil Logol, ce qui a permis de faire évoluer le logiciel.
Nous avons notamment été confronté à la nécessité de pouvoir poser des contraintes sur des séquences discontinues. En effet, le taux de GC de la tige est calculé sur la somme des deux parties de la tige. Initialement, il n'était possible d'appliquer ce genre de contrainte que sur des segments continus : sur une « variable » (sens Logol du terme) ou sur une « vue ».
Une nouvelle version de l'outil Logol intègre désormais cette possibilité de poser des contraintes globales sur un ensemble de segments discontinus. Ces contraintes sont gérées en tant que post-traitement. Elles seront posées globalement dans une table de calcul, au même niveau que la table de définition de morphismes. Dans le cas d'un modèle Logol sous forme grammaticale, elles sont mises avant les règles de grammaire. Dans le cas d'un modèle Logol sous forme graphique, elles seront enregistrées dans la table de calcul. Nous avons également pointé les limitations d'autres post-traitements existant qui bornent le nombre de matchs retournés en résultat. Celui-ci étant appliqué avant le dernier filtre implanté, nous avons volontairement choisi un nombre maximal de résultats très supérieur (1000) au nombre de résultats espérés.
Pour le choix final du modèle Logol (cf section 3.1.4), lors du test des deux modèles, le taux de GC n'a pas été ajouté dans les contraintes des modèles Logol, mais testé en post-filtre (cf section 5.3). En effet le temps d'exécution de l'analyse Logol étant assez long, il s'est avéré moins coûteux en temps de réaliser un modèle très large puis d'appliquer un post-filtre sur les matchs lors de l'analyse des résultats au lieu d'utiliser les contraintes dans Logol et de relancer l'analyse Logol.
Lors de la mise à jour les annotations de tige-boucles de la base Crispi, l'analyseur Logol est par défaut lancé de telle manière qu'il se limite à un job et confronte les séquences au modèle les unes après les autres. Il existe cependant une option dans la ligne de commande permettant que chaque séquence soit analysée dans des jobs séparés en parallèle. La taille très réduite des séquences et leur grand nombre réduisent alors énormément le temps d'analyse (de plusieurs heures à environ un

quart d'heure). En phase de test, cette option n'est pas toujours utilisée, notamment pour éviter de surcharger trop souvent le cluster.

5.3 Traitement des résultats issus de l'analyseur Logol

Les résultats retournés par Logol sont sous la forme d'une archive zip contenant, pour chaque séquence qui a été confrontée au modèle, un fichier au format XML. Le nombre de séquences à analyser est trop importante pour pouvoir analyser les résultats à la main. Il existe un outil accessible en ligne, sur le site de Genouest, mais il permet uniquement d'explorer les résultats de manière « User Friendly ». Il ne permet pas de réaliser des calculs et de faire une synthèse d'informations sur l'intégralité des résultats. Il a donc été nécessaire de réaliser un outil permettant d'analyser les résultats délivrés par l'analyseur Logol. Plusieurs critères sont apparus importants. Premièrement, le temps de traitement doit rester dans la mesure du possible assez bas pour réaliser des calculs sur l'ensemble des résultats. De plus, du fait que la succession des éléments du fichier XML dépend de la grammaire employée, le système permettant la conversion du XML en un objet tige-boucle doit être facile à modifier.
XML possède un large panel d'outils associés, mais aucun ne permet de répondre à tous ces critères. C'est donc vers Java que le choix s'est porté. Java est un langage facile à utiliser avec des performances correctes. En plus d'un JDK bien fourni, il possède un grand nombre de bibliothèques indépendantes. Parmi les bibliothèques du JDK, il y a SAX et DOM, deux outils pour analyser un fichier XML. Cependant, il existe une bibliothèque indépendante permettant de parcourir l'arbre XML de manière naturelle, JDOM (http://www.jdom.org/). JDOM utilise un analyseur XML, il peut utiliser aussi bien SAX que DOM, ainsi que d'autres analyseurs. En plus d'une très grande simplicité d'utilisation, l'outil reste très efficace. Une première version de l'analyseur assez simple à été rapidement développée. Il a permis d'analyser 10000 résultats en un temps de traitement inférieur à la minute sachant que le programme était exécuté sur l'ordinateur local et les données placées sur un disque réseau, temps tout à fait acceptable pour réaliser les analyses de résultats complets. L'outil a donc été développé à nouveau de manière qu'il corresponde aux critères énoncés précédemment. Réaliser un couplage entre structures et fonctionnalités n'est pas apparu probant. En effet, ici, il est assez difficile à concevoir qu'un objet « tige-boucle » soit lié à une manière de l'extraire d'un document XML, surtout que comme il sera vu plus tard, ce même objet pourra être extrait d'une expression parenthésée. Dans certains langages impératifs, il existe des outils pour faire d'une certaine façon du fonctionnel. En C, par exemple, il existe les pointeurs de fonction, en C++ il existe en plus les foncteurs, des classes dont l'opérateur '()' est surchargé. En java, même si rien de tout ceci n'existe, il est assez simple de reprendre l'idée de la surcharge de l'opérateur '()' en utilisant simplement un méthode dédiée pour l'aspect fonctionnel. Associé à une fabrique qui instancie dynamiquement les objets, il est assez aisé de modifier les fonctions utilisées tant que les fonctions respectent une interface. Ainsi la réalisation de l'analyseur a été découpée en un ensemble de tâches unitaires. Nous avons cherché à réaliser un outil le plus générique possible en retardant au plus tard la spécialisation des fonctions pour résoudre le problème concernant l'analyse des tige-boucles. Ce choix a été fait pour pouvoir réduire le travail à chaque modification du modèle et éventuellement réutiliser dans d'autres programmes ces classes. Afin de rendre le programme le plus dynamique possible, les chaînes de caractères et un certain nombre de propriétés ont été externalisées dans des fichiers de propriétés compatibles avec la classe du JDK « java.util.Properties ». Au final, cet analyseur termine l'analyse des résultats de plus de 30 000 fichiers enregistrés sur un disque réseau en des temps inférieurs à cinq minutes, temps tout à fait acceptables en pratique. De plus, avec la possibilité d'ajouter des filtres, il est possible d'éviter d'avoir à lancer l'analyseur Logol pour chaque modification sur les contraintes d'un modèle.

5.4 Mise en œuvre de la procédure de mise à jour de la base

5.4.1 Implémentation

L'application effectuant la mise à jour de l'annotation des tiges-boucles dans la base CRISPI est divisée en quatre étapes successives qui communiquent par fichiers. Chaque étape est clairement séparée des autres, les étapes ne communiquant entre elles que par fichiers. De plus, il est nécessaire, lors de l'étape d'analyse Logol, de pouvoir au moins lancer une commande. De ce fait, chaque étape sera réalisée par un exécutable indépendant et l'exécution de chaque étape sera faite à l'aide d'un script « bourne shell ». L'étape de l'analyse Logol ne nécessite qu'une ligne de commande. Pour les autres étapes, l'exécutable sera codé en Java.
Nous avons repris l'architecture d'objet-fonction de l'analyseur de résultats pour les mêmes raisons qui ont poussé à utiliser cette architecture pour ce dernier. Nous avons aussi conservé l'utilisation des fichiers de propriétés. L'analyseur a été lui aussi réutilisé et intégré à la dernière étape. Il a été modifié afin qu'il puisse partir directement de l'archive zip.
L'interfaçage de Java avec les bases de données peut être réalisé de diverses manières. Le plus simple dans le cas actuel est d'utiliser les interfaces JDBC pour travailler avec la base de données MySQL.

5.4.2 Déploiement du traitement sur le site opérationnel

Le code source et tous les fichiers nécessaires à l'exécution du script sont placés dans un dépôt SVN. Le code source est commenté, mais aussi documenté à l'aide des balises Javadoc. Une documentation sur la manière de déployer l'application, les différentes propriétés dans chaque fichier de propriétés, ainsi qu'une rapide explication sur l'utilité de chaque classe, est réalisée.
Un script ANT est écrit afin de permettre un déploiement automatique à partir des sources. Les sources sont ainsi compilées, les jars sont créés à partir des sources et placés dans un dossier donné. Les fichiers nécessaires, comme les jars externes (JDOM et les pilotes MySQL), les scripts shell ou encore les fichiers de propriétés, sont copiés dans le même dossier que les jars créés précédemment. Chaque jar exécutable est associé à un petit script shell qui charge l'environnement Java sur le cluster et exécute le jar. Ainsi, chaque étape peut être exécutée indépendamment des autres. Ne disposant pas statiquement le dossier de destination, il a fallu utiliser des chemins relatifs. L'une des premières actions de chaque script shell est donc de placer le dossier courant à l'endroit où le script est enregistré. Ce chemin est trouvé par la commande scriptPath="$(cd "$**(dirname** "$0" **)**" && pwd)"**)**
L'automatisation implique des contraintes. Toute erreur doit être remontée et doit être signalée dans la sortie « erreur ». Même si l'ensemble du code a été testé avec succès, il n'est pas possible d'affirmer qu'aucune exception ne sera levée. En plus de potentielles erreurs dans le code, il faut aussi prévenir des erreurs de l'utilisateur, ou plutôt ici des erreurs de configuration. Nous avons essayé de prévenir au plus tôt ce type d'erreur. Cependant la manipulation des paramètres n'est pas aisée dans un script bourne shell. Il n'est donc pas encore possible de proposer un système de paramètres souples pour le moment. La difficulté vient de la nécessité de pouvoir lancer une ligne de commande (pour exécuter l'analyseur Logol). Parmi les solutions possibles permettant une meilleure gestion des paramètres, il y a la réécriture du script principal en Perl. Cela permettrait aussi d'utiliser la bibliothèque Perl Schedule-DRMAAc (http://search.cpan.org/dist/Schedule-DRMAAc/) pour réaliser une division du travail en plusieurs jobs sur le cluster, en particulier pour l'analyse Logol. Nous avons récemment également remarqué la possibilité d'exécuter une ligne de commande à l'intérieur de Java à l'aide de la classe « java.lang.Runtime ».