# Algorithmique pour la recherche de motifs approchée et application à la recherche de cibles de microARN

Christophe Vroland

Université Lille Nord de France
Pôle de Recherche et d'Enseignement Supérieur

Université de Lille 1 SCIENCES ET TECHNOLOGIES

**Université Lille I**

Écoles doctorales **ED Régionale SPI 72, SMRE 104**

Unité de recherche **CRIStAL – UMR CNRS 9189 Université Lille 1**

Thèse présentée par **Christophe Vroland**

Soutenue le **18 mai 2016**

En vue de l'obtention du grade de docteur de l'Université Lille I

Discipline **Informatique**

# Algorithmique pour la recherche de motifs approchée et application à la recherche de cibles de microARN

| | | |
|---|---|---|
| **Thèse dirigée par** | Hélène Touzet | directrice |
| | Vincent Castric | co-directeur |
| | Mikaël Salson | co-encadrant |

**Composition du jury**

| | | |
|---|---|---|
| *Rapporteurs* | Guillaume Blin | professeur à l'Université de Bordeaux |
| | Pierre Peterlongo | chargé de recherche HDR à l'Inria – Centre Inria Bretagne Atlantique |
| *Examinateur* | Maud Tenaillon | directrice de recherche au CNRS – UMR de Génétique Quantitative et Évolution – Le Moulon |
| *Directeurs de thèse* | Hélène Touzet | directrice de recherche au CNRS – CRIStAL ; Inria Lille - Nord Europe |
| | Vincent Castric | chargé de recherche HDR au CNRS – EEP |
| | Mikaël Salson | MCF à l'Université Lille 1 - CRIStAL ; Inria Lille – Nord Europe |

**Mots clés :** bioinformatique, algorithmique du texte, recherche de motifs approchée, microARN, *arabidopsis*, régulation de gène

**Keywords:** computational biology, text algorithms, approximate string matching, microRNA, *arabidopsis*, gene regulation

Cette thèse a été préparée dans les laboratoires suivants.

**CRIStAL – UMR CNRS 9189 Université Lille 1**

Cité scientifique - Bâtiment M3 extension
Avenue Carl Gauss
59655 Villeneuve d'Ascq Cédex
France

Site `http://www.cristal.univ-lille.fr/`

**Bonsai – Inria Lille Nord Europe**

Parc scientifique de la Haute Borne
40, avenue Halley - Bât A - Park Plaza
59650 Villeneuve d'Ascq
France

Site `http://www.inria.fr/equipes/bonsai`

**Évo Éco Paléo – UMR CNRS 8198 Université Lille 1**

Cité scientifique – Bâtiment SN2
59655 Villeneuve d'Ascq Cédex
France

Site `http://gepv.univ-lille1.fr/`

Une thèse n'est pas une aventure qui se fait seul et si j'ai réussi jusqu'ici, c'est que j'ai reçu de l'aide de nombreuses personnes.

Si je suis arrivé jusqu'ici, c'est avant tout grâce à mes parents qui m'ont permis de réussir dans mes études, ils m'ont encouragé, aidé et permis de m'investir pleinement dans celles-ci. Ils m'ont suivi et soutenu durant tout mon parcours.

Je suis arrivé en informatique un peu par hasard. J'ai eu la chance d'avoir eu très tôt accès à un ordinateur, j'ai beaucoup appris par moi même par passion, mais je n'avais jamais imaginé y faire mes études. C'est lors de mon stage en Master 2 au sein de l'UMR STLO à Rennes, alors que j'étais étudiant en microbiologie, que Nadia Berkova avait remarqué ma passion pour l'informatique et me poussa à m'intéresser à la bioinformatique. Et alors que j'étais en milieu de mon dossier d'inscription dans un autre master de biologie pour élargir ma voie, me voici en train de postuler in extremis dans un master de compétence complémentaire en informatique.

L'autre événement particulièrement fou dans mon parcours, c'est que malgré mon profil, trois personnes ont pris le pari insensé de me donner la chance de réaliser cette thèses en informatique. Hélène, Vincent et Mikaël, là dessus, un grand merci.

Mais arriver en thèse, ce n'est que le début du cette épreuve, et il faut beaucoup de soutien et d'encouragement de la part de ses encadrants, de ses collègues, de sa famille et de ses amis. Tous ont participé à leur façon à la réalisation de cette thèse. Bien sûr encore une fois, mes parents et mes encadrants ont été parmis les acteurs les plus directs. À mes collègues de bureaux qui ont dû me supporter, désolé. J'ai bien aimé discuter avec vous, Tu, Mohcen Thierry, Yoann, Tatiana, Valentin, Benjamin, Laura, Nicolas, Anne. Il y avait aussi ces pauses café interminables où l'on parlait de tout, parfois de boulot, parfois de loisir. C'est lors de ces pauses que j'ai eu l'occasion de discuter plus longuement avec Laurent et ces discussion m'ont grandement inspiré pour avancer dans mon travail. Je n'oublie pas non plus ces trois semaines passées dans le laboratoire de Barbara Baker, une personne formidable avec qui il a été très intéressant de parler.

Il y a aussi mes amis, Delphine, Guillaume, Randolph, Damien et tous les autres, ces moments passés avec vous m'ont permis de penser à autre chose que le travail.

Mais dans tout ce parcours, les derniers pas sont les plus marquants. Je remercie Maud Tenaillon, Guillaume Blin et Pierre Peterlongo d'avoir accepté de faire partie de mon jury et malgré les conditions très particulières de ma soutenance, vous aussi, vous vous êtes adaptés. Vos remarques étaient très intéressantes. Enfin lors de ma soutenance, alors que je paniquais, je me suis souvenu d'une discussion que j'avais eu au début de ma thèse avec Evguenia le jour de sa soutenance. Je lui avais fait part de ma surprise de son grand calme et

elle me répondit alors qu'elle n'avait aucune raison particulière de stresser, que tout était prêt. C'est alors qu'une fois devant le tableau que je me suis rappelé de cette remarque, les membres du jury était tous là en visioconférence, mes encadrants en face, mes diapos faites et mon texte, je le connaissais..., tout était prêt. Il ne restait plus qu'à savourer le fruit de ces années de thèse.

Je ne dirais pas qu'une thèse ce n'est que du bonheur, loin de là, c'est énormément d'épreuves et de difficultés. Mais c'est une fois qu'on les a dépassées que l'on savoure ce bonheur. Et si j'ai réussi à les dépasser, c'est grâce à toutes ces personnes que j'ai croisées durant toutes ces années.

A tout ce monde et à tous ceux que je n'ai pas pu citer, un grand merci.

**Algorithmique pour la recherche de motifs approchée et application à la recherche de cibles de microARN**

**Résumé**

La recherche de motifs approchée consiste à identifier les occurrences d'un motif modulo une certaine distance au sein d'un texte. Ce problème trouve de nombreuses applications en bio-informatique pour l'analyse de séquences biologiques. Par exemple, les microARN sont des petits ARN qui régulent l'expression des gènes par reconnaissance d'un motif similaire. Comprendre le mode d'action des microARN demande de pouvoir localiser de courts motifs, environ 21 nucléotides, comprenant jusqu'à 3 ou 4 erreurs dans un texte de l'ordre de $10^8$ à $10^9$ nucléotides, représentant un génome. Dans cette thèse, nous proposons un algorithme efficace pour la recherche de motifs approchée, qui se base sur la définition d'un nouveau type de graines avec erreurs, les graines $01^*0$, et qui exploite une structure d'index compressée, le FM-index. Cet algorithme a été mis en œuvre dans un logiciel librement disponible, appelé Bwolo. Nous démontrons expérimentalement l'avantage de cette approche en nous comparant à l'état de l'art des outils existants. Nous montrons également comment utiliser Bwolo pour mettre en place une analyse originale sur l'étude de la distribution des cibles potentielles de miARN dans deux génomes de plantes, *Arabidopsis thaliana* et *Arabidopsis lyrata*.

**Mots clés :** bioinformatique, algorithmique du texte, recherche de motifs approchée, microARN, *arabidopsis*, régulation de gène

**Abstract**

Approximate string matching consists in identifying the occurrences of a motif within a text, modulo a given distance. This problem has many applications in bioinformatics for the analysis of biological sequences. For instance, microRNAs are short RNA molecules regulating the expression of genes by specific recognition of their sequence motif on the target gene. Understanding the mode of action of microRNAs requires the ability to identify short motifs, around 21 nucleotides in size, comprising up to 3-4 errors in a text whose size is in the order of $10^8$-$10^9$, representing a genome. In this thesis, I have proposed an efficient algorithm for the approximate search of short motifs. This algorithm is based on a new type of seeds containing errors, the $01^*0$ seeds, and uses a compressed index structure, the FM-index. I have implemented this algorithm in a freely available software, Bwolo. I demonstrate experimentally the advantage of this approach and compare it to the state of the art of existing tools. I also show how Bwolo can be used and have set up an original study on the distribution of potential miRNA target sites in two plant genomes, *Arabidopsis thaliana* and *Arabidopsis lyrata*.

**Keywords:** computational biology, text algorithms, approximate string matching, microRNA, *arabidopsis*, gene regulation

**CRIStAL – UMR CNRS 9189 Université Lille 1**
Cité scientifique - Bâtiment M3 extension – Avenue Carl Gauss – 59655 Villeneuve d'Ascq Cédex – France

# Liste des tableaux

# Table des figures

# Introduction Générale

Il est aujourd'hui devenu particulièrement aisé de séquencer un génome. Depuis l'époque des premiers séquençages, la technologie a significativement évolué, aussi bien en termes de coût que de débit. Une fois la séquence d'un génome obtenue, il reste toutefois à l'exploiter. De nombreuses informations peuvent être déduites, comme la localisation des gènes, les mutations, la phylogénie... L'analyse de ces séquences se fait à l'aide d'outils bioinformatiques qui travaillent sur des données textuelles. Cependant, cette exploitation fait face au problème de la taille ou au nombre important des séquences à considérer. La taille d'un génome varie de quelques millions à plusieurs milliards de nucléotides. De même, le nombre de séquences produites par un séquenceur se compte en millions voire en milliards. La recherche d'information dans cette quantité de données demande des outils extrêmement efficaces et en constante évolution. Il existe par exemple une centaine d'outils pour la recherche de courts fragments dans un génome (outils de *mappings*) [1]. Pour de multiples raisons (erreurs de séquençage, mutations...), la recherche d'un motif dans un génome nécessite souvent de tolérer des différences. D'un point de vue algorithmique, cela revient au problème de la recherche de motifs approchée dans un texte. Cette problématique est assez ancienne. Parmi les tous premiers algorithmes publiés, celui de Needleman et Wunsch date du début des années 1970 [118]. Les méthodes actuelles visent à améliorer les performances en temps, en mémoire et à tolérer un plus fort taux d'erreurs.

Parmi les problèmes spécifiques de recherche de motifs approchée, celui de la recherche de motifs courts avec un fort taux d'erreurs reste un des plus

1. http://www.ebi.ac.uk/~nf/hts_mappers/

difficiles. Or il existe un mécanisme en biologie pour lequel ce genre de solution est nécessaire : les cibles de microARN (miARN). Les miARN sont de petits ARN d'une taille d'environ 21 nucléotides qui régulent l'expression de nombreux gènes en ciblant leur séquence codante par complémentarité imparfaite de séquences (environ trois différences). Des modes d'action découverts récemment, comme l'existence de cibles en amont ou dans les introns d'un gène, nécessitent de rechercher les cibles de miARN non plus dans les seules séquences des gènes, mais dans l'intégralité du génome. Cette recherche demande d'identifier des courtes séquences avec un taux d'erreurs avoisinant les 15% dans une séquence de l'ordre de $10^8$ à $10^9$ nucléotides. D'un point de vue algorithmique, il s'agit donc de rechercher de courts motifs à trois erreurs dans de longues séquences, sur un alphabet composé de quatre lettres A, C, G et T. Dans cette thèse je m'intéresserai à ce problème et je montrerai comment l'appliquer à la problématique biologique des miARN et de la recherche de leurs cibles.

Le document est organisé en trois chapitres. Dans le chapitre 1, je ferai un résumé sur la prédiction de cibles de miARN en commençant par un rappel des notions de base en biologie moléculaire, pour arriver ensuite à la description des miARN et enfin décrire comment la recherche de cibles de miARN est traitée en bioinformatique, de la description d'un modèle de cible jusqu'à sa mise en application. Dans le chapitre 2 est abordée la problématique de la recherche de motifs approchée. Après une présentation de l'état de l'art et des index plein texte, je définirai un nouveau type de graine, appelé $01^*0$, qui est spécialement adapté à la recherche de petits motifs en acceptant un fort taux d'erreurs. Ces graines sont mises en œuvre dans le logiciel Bwolo. Enfin, dans le chapitre 3, nous expliquerons comment utiliser Bwolo dans le contexte d'une étude à grande échelle de la distribution des cibles potentielles de miARN dans les génomes de *Arabidopsis thaliana* et *Arabidopsis lyrata*, deux plantes du genre *Arabidopsis*.

Chapitre 1

# Le contexte biologique

Le travail de cette thèse porte sur la compréhension du mode d'action de petits éléments génétiques, les microARN (miARN). Les miARN ont été identifiés relativement récemment, et constituent une exception au modèle «classique» de la biologie moléculaire, qui a acquis le statut de paradigme voire de «dogme» comme formulé par Crick en 1970 [25]. D'un point de vue bioinformatique, la contribution de ce travail repose sur le développement et l'implémentation de méthodes d'algorithmique du texte. Ces deux domaines d'étude faisant appel à des notions issues de communautés scientifiques distinctes, je présente dans ce chapitre la question biologique sous la forme d'entités manipulables par des informaticiens. Je vais donc ici présenter un modèle simplifié destiné à permettre la compréhension de la suite du document. J'essaierai également d'apporter au fur et à mesure les détails nécessaires.

Dans un premier temps, je vais présenter un modèle simplifié de la biologie moléculaire (aussi appelé «Dogme central de la biologie moléculaire» [25]) qui décrit comment il est possible de passer de l'ADN aux protéines et aux caractères qu'elles contrôlent. J'apporterai plus de précision à ce modèle pour y intégrer les miARN. Ce faisant, nous serons confrontés au flou de certaines définitions. Cela reflète le fait que la biologie est une science expérimentale, qui progresse par observations. Des hypothèses sont émises, puis vérifiées ou réfutées par des expériences. Les objets biologiques issus de cette démarche sont des concepts, des catégories dont les définitions précises sont amenées à évoluer en fonction des nouvelles observations. Ces concepts et ces catégories sont des abstractions qui sont nécessaires pour appréhender le vivant. Le domaine des miARN, relativement jeune, est particulièrement caractéristique de ce processus en cours.

## 1.1 La théorie fondamentale de la biologie moléculaire

Le *dogme central de la biologie moléculaire* explique le circuit de l'utilisation de l'information génétique au sein de la cellule. Il fait intervenir trois types de séquences : l'ADN (acide désoxyribonucléique), l'ARN (acide ribonucléique) et

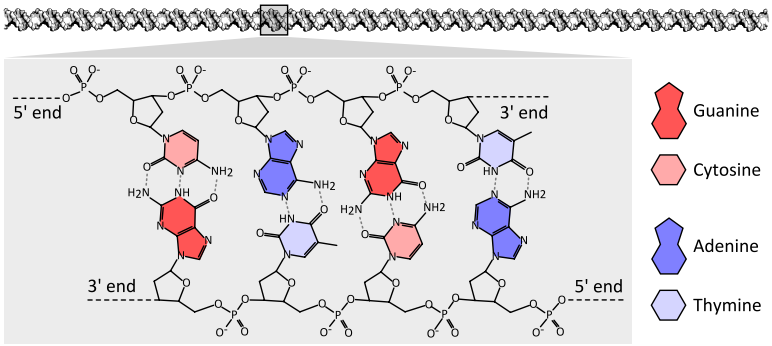



FIGURE 1.1 – Structure de l'ADN. La double hélice d'ADN est composée de deux brins. Chaque brin est un assemblage linéaire de nucléotides (Guanine, Cytosine Adénine, Thymine). La suite de nucléotides d'un brin est dépendante des nucléotides de l'autre brin suivant les règles d'appariement A-T et G-C. Les brins sont orientés. A titre d'exemple, la séquence représentée ci-dessus se lit CAGT ou ACTG selon le brin considéré. (source : `wikimedia.org`)

les protéines.

### 1.1.1 L'ADN

L'ADN est le support de l'information génétique chez les êtres vivants. Cette information génétique est héréditaire : nous la recevons de nos parents, qui l'ont eux-mêmes reçue de leurs parents et ainsi de suite.

#### Une séquence ...

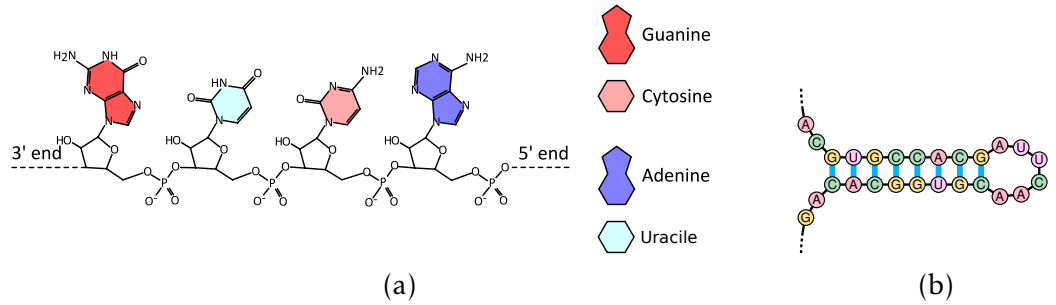
L'ADN est une macromolécule biologique qui, avec l'ARN, forme la famille des acides nucléiques. Il est caractérisé par la formation de deux brins enroulés l'un autour de l'autre pour former une double hélice. L'ADN est un polymère linéaire, c'est-à-dire une répétition de sous-unités en une seule chaîne sans embranchements. Dans le cas des acides nucléiques, ces sous-unités sont les *nucléotides*. Dans l'ADN, quatre nucléotides différents sont utilisés qui sont distingués par leur partie variable, la base azotée, l'adénine (A), la cytosine (C), la guanine (G) et la thymine (T). L'enchaînement de ces différents nucléotides forme une séquence que l'on peut représenter par une chaîne de caractères sur l'alphabet {A, C, G, T}.

La molécule d'ADN est orientée et possède une extrémité dite 5' et une autre dite 3'. Ce chiffre correspond au numéro du carbone dans la structure chimique. On peut donc donner un sens à la séquence. Conventionnellement, une séquence est écrite à partir de son extrémité 5' vers son extrémité 3'. Dans la structure en double hélice, les deux brins sont dans des sens opposés, on dit qu'ils sont *anti-sens*.

Un aspect important de la structure en double hélice est la complémentarité des nucléotides. En effet, les acides nucléiques sont caractérisés par une complémentarité des bases azotées. Ainsi, tels les deux pôles des aimants, les bases s'apparient deux à deux selon certaines règles au travers de liaisons hydrogène. En face d'une adénine (A) se trouvera une thymine (T) associée par 2 liaisons hydrogène et vice-versa. De même, en face d'une cytosine (C) se trouvera une guanine (G) associée par 3 liaisons hydrogène. Les liaisons hydrogène entre deux nucléotides contribuent à la stabilité de la structure en double hélice et de manière plus générale de toutes les structures en double brin. Grâce à cette complémentarité, un seul des deux brins est nécessaire pour décrire l'ensemble de l'information contenu dans les deux brins. Pour retrouver la séquence du brin opposé, il suffit de changer les nucléotides par leur nucléotide complémentaire et d'inverser la séquence ainsi obtenue. On dit que le second brin est le complémentaire inverse.

### . . . support de l'information génétique

Toujours selon le dogme central, l'information génétique est portée par les *gènes* qui en constituent l'unité de base. Le concept de gène lui-même a connu de multiples évolutions au cours de l'histoire de la génétique. Dans le cadre de ce travail, nous adopterons une définition restrictive basée sur la nécessité que cette séquence doit être au moins transcrite en ARN et code pour une unité fonctionnelle à part entière. L'ensemble de l'ADN d'un individu est appelé *génome*. Toutes les cellules d'un individu possèdent le même génome, bien que quelques événements particuliers soient capables d'altérer celui-ci (tels que les mutations somatiques [16] ou la recombinaison V(D)J [161]) et que nous ne considérerons donc pas ici. La taille d'un génome est très variable




(a) (b)

FIGURE 1.2 – (a) Structure simple brin de l'ARN. L'ARN est très similaire à l'ADN. (b) Structure en tige-boucle de l'ARN. L'ARN est capable de se replier sur lui-même pour former des structures secondaires.(source : `wikimedia.org`)

selon les espèces. Celui de l'Homme (*Homo sapiens*) par exemple fait environ $3{,}4 \cdot 10^9$ nucléotides. Celui de la levure de bière (*Sacchoromyces cerevisiae*) fait seulement $12 \cdot 10^6$ nucléotides, mais celui du maïs (*Zea mays*) est plus grand et fait $2{,}4 \cdot 10^9$ nucléotides. Les plus petits génomes connus (en excluant ceux des virus) sont retrouvés chez des bactéries endosymbiotiques, avec par exemple *Nasuia deltocephalinicola* qui détient l'actuel record avec seulement 112 091 nucléotides (soit plus de 30 000 fois plus petit que le génome humain). À l'inverse, les plus grands génomes connus sont retrouvés chez des unicellulaires avec par exemple chez *Polychaos dubium*, une amibe, qui possède un génome de $675 \cdot 10^9$ nucléotides (soit presque 200 fois plus grand que le génome humain)[115]. D'une façon générale la quantité d'information contenue dans le génome en terme d'arrangements de nucléotides est considérable. En terme algorithmique, un génome peut être considéré comme un texte de très grande taille.

### 1.1.2 L'ARN

L'ARN est très similaire à l'ADN (Figure 1.2a). C'est également un acide nucléique qui prend la forme d'un polymère linéaire. Cependant, en plus d'une légère modification dans la composition chimique du squelette (le sucre a un atome d'oxygène en plus étant donc un ribose au lieu d'un désoxyribose dans l'ADN), il existe une autre différence au niveau des bases azotées. En effet la thymine (T) dans l'ADN est remplacée par l'uracile (U) dans l'ARN. Ces deux

bases possèdent les mêmes propriétés de complémentarité avec l'adénine (A). Cependant l'uracile est capable de s'apparier, dans une moindre mesure, avec la guanine (G), tandis que dans l'ADN une interaction G-T est impossible. Une autre différence importante est que l'ARN est simple brin. Ce brin d'ARN résulte de la copie de l'un des deux brins de l'ADN au niveau d'un gène. Selon le dogme central, l'ARN est considéré comme une simple copie intermédiaire de l'information génétique et n'a pas d'autre fonction dédiée que de transporter cette information de l'ADN contenu dans les chromosomes vers le reste de la cellule.

Une autre propriété fondamentale de l'ARN pertinente dans le cadre de cette thèse, est sa capacité à s'apparier à lui-même. En effet, l'ARN étant simple brin, les bases azotées sont libres de s'apparier entre elles selon les même règles de complémentarité que pour l'ADN, la thymine étant remplacée par l'uracile. De ce fait, si un brin d'ARN possède deux sous-chaînes complémentaires inverses l'une de l'autre, ces deux sous-chaînes vont avoir tendance à s'apparier ensemble. Il y a formation d'une structure dite en "tige-boucle" composée d'une section double brin formée par les deux sous-chaînes appariées (la tige) et d'une section simple brin formée de la partie comprises entre ces sous-chaînes (la boucle, voir Figure 1.2b). Ainsi, l'ARN va avoir la capacité à former des structures dites secondaires dont les fonctions sont essentielles dans la cellule. Cette propriété peut être étendue à deux brins d'ARN si chacun possède une sous-chaîne qui est la complémentaire inverse de l'autre. Ces deux brins vont alors pouvoir de la même façon s'apparier.

### 1.1.3 Les protéines

Les protéines sont une troisième classe de macromolécules biologiques. Cependant, les sous-unités qui les constituent ne sont pas des nucléotides, mais une toute autre famille de résidus, les acides aminés. Dans le cas des protéines, il existe 20 acides aminés différents qui servent de briques élémentaires.

Par le jeu d'interactions physico-chimiques entre les résidus et aussi avec l'environnement, la chaîne d'acides aminés va se replier sur elle-même. Il en résulte la formation de structures tridimensionnelles complexes. Les formes

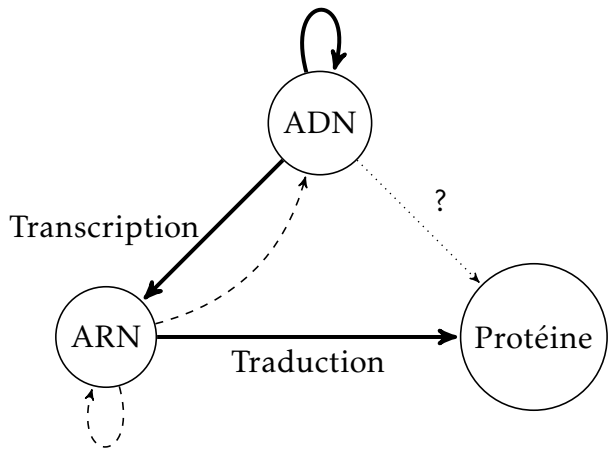



FIGURE 1.3 – Le dogme central de la biologie moléculaire selon CRICK [25]. L'ADN est transcrit en ARN qui est traduit à son tour en protéine. CRICK avait aussi proposé comme théoriquement possible les transitions entre l'ARN vers l'ADN, l'ARN vers l'ARN et l'ADN vers la protéine. Les deux premières ont été mises en évidence mais restent exceptionnelles (tirets) alors qu'aucune observation n'a été faite à ce jour avérant de l'existence de la troisième (pointillé).

de ces structures vont permettre l'interaction avec d'autres molécules. Ainsi, il sera possible à certaines protéines de s'associer entre elles pour avoir un rôle structural. D'autres vont aussi pouvoir catalyser une réaction chimique. Ces protéines ayant un rôle de catalyseur sont appelées enzymes. Il est courant qu'une protéine n'agisse pas seule, mais qu'elle fasse partie d'un complexe enzymatique.

Dans le cadre du dogme central, les protéines sont les macromolécules qui sont actives. Au travers de leurs activités, elles vont être responsables de l'ensemble des caractères portés par un individu (son phénotype).

### 1.1.4 De l'ADN à l'ARN, aux protéines, au phénotype

Les différents acteurs ayant été présentés, je vais détailler les processus qui permettent à l'information génétique de passer de l'un à l'autre : la *transcription*, la *traduction* puis l'interaction des protéines produites au sein de *réseaux*.

### La transcription

La *transcription* consiste en la copie d'une séquence d'ADN en une molécule d'ARN appelé ARN pré-messager. Cette étape est réalisée entre autres par une enzyme, l'ARN polymérase II (pol II) responsable de la polymérisation de l'ARN en prenant comme matrice de copie l'un des deux brins d'ADN du gène. L'ARN polymérase II est recrutée à l'aide de signaux présents dans le *promoteur* du gène qui sont généralement situés à moins d'un millier de nucléotides en amont du site d'initiation de la transcription. Le brin matrice est appelé brin transcrit. Pour un gène donné, le brin transcrit sera toujours le même. La sélection des nucléotides à ajouter se fait grâce à la complémentarité des bases. L'ARN pré-messager est donc le complémentaire inverse du brin transcrit. Par construction, il est la copie du brin opposé, dit brin codant.

Cette molécule d'ARN n'est pas encore mature et va subir un ensemble de modifications, dont une étape d'épissage. Durant cette étape, des régions, appelées introns, sont éliminées. Les régions qui ont été conservées sont appelées *exons*. Les molécules d'ARN vont subir encore plusieurs d'autres modifications incluant l'ajout d'une coiffe 7-methylguanosine en 5' et l'ajout d'une queue de polyA (une répétition de A) en 3' qui leur permettent d'éviter la dégradation par un ensemble de mécanismes cellulaires. Après ceci, on obtient un ARN messager mature, ou ARNm, qui est exporté hors du noyau vers le cytoplasme.

### La traduction

L'ARNm va maintenant servir de support pour la dernière étape, la traduction. La traduction permet de produire des protéines à partir de l'ARN. Il y a un changement de langage : d'une chaîne d'acides nucléiques d'un alphabet de taille 4 en une chaîne d'acides aminés d'un alphabet de taille 20. Cette traduction se fait au travers d'un code quasi universel, nommé code génétique. Celui-ci se base sur des triplets de nucléotides qui vont chacun correspondre à un acide aminé. Ces triplets sont appelés *codons*. Sachant qu'il y a 4 nucléotides possibles, il y a $4^3 = 64$ codons possibles. Il existe 3 codons STOP codant pour l'arrêt de la traduction et un codon START. Mais il y a seulement 20 acides aminés. Le code génétique possède donc de nombreuses redondances.

La traduction est réalisée par un complexe enzymatique, les *ribosomes*. Ils vont «lire» codon par codon, sans chevauchement ni décalage, et ajouter à la chaîne d'acides aminés en construction l'acide aminé associé au codon en cours.

L'initialisation de la traduction se fait toujours au même endroit, sur le même *cadre de lecture* en commençant par le premier codon START (AUG) présent sur l'ARNm. La terminaison de la traduction, quant à elle, est dirigée par la lecture de l'un des trois codons STOP (UAA, UAG, UGA). La partie ainsi traduite ne couvre pas l'intégralité de la séquence de l'ARNm. Il existe donc des régions, au début et à la fin de l'ARNm, qui ne sont pas traduites en protéine. Elle sont appelées respectivement 5'UTR et 3'UTR (UTR : u*ntranslated* r*egion*) selon leur localisation par rapport aux extrémités 5' ou 3' de l'ARNm. La protéine formée peut finalement subir quelques modifications *a posteriori* avant d'être active, telles que l'ajout de sucres (glycolysation), la formation de ponts disulfures, la formation de structures quaternaires composant des complexes protéiques…

**Réseaux de régulation de l'expression**

Nous venons d'expliquer comment il est possible de passer d'un gène à un caractère en passant par un ARN messager. Cependant, ce modèle est très incomplet et nécessite d'introduire une notion de régulation. En effet la majorité des gènes s'expriment sous le contrôle de différents facteurs, en particulier dans le contexte de la différenciation cellulaire. Ainsi, si les fonctions de base de toutes les cellules sont partagées, la majorité des cellules est largement différenciée, en lien avec les différences d'expression de leur répertoire de gènes. À titre d'exemple, myocyte (une cellule de la fibres musculaires) ne va pas produire le même répertoire de protéines que celui qui est produit par un hépatocyte (une cellule du foie). De la même façon, de nombreux gènes ne sont activés que dans certaines conditions environnementales, qu'elles soient biotiques ou abiotiques.

La régulation peut être réalisée à chaque étape du modèle présenté ici, selon un très grand nombre de mécanismes qui ne sont par encore totalement connus. Ainsi, nous allons distinguer quatre types de régulation en fonction de l'étape à laquelle elles agissent : la régulation transcriptionnelle, la régulation post-transcriptionnelle, la régulation traductionnelle ou la régulation

post-traductionnelle. Par exemple, sur l'ADN, les séquences en amont ou en aval d'un gène contiennent des signaux permettant la promotion ou l'inhibition de la transcription par le recrutement de régulateurs transcriptionnelles. Les premiers régulateurs décrits ont été les facteurs de transcription, qui sont des protéines ayant la propriété de se fixer sur des séquences d'ADN et permettant le recrutement de la machinerie de transcription [150].

Ces systèmes peuvent adopter des architectures très différentes. Dans certains cas, la régulation peut consister à la simple régulation d'un seul gène par un régulateur. Dans d'autres cas, un même régulateur peut contrôler toute une série de gènes différents (il est dit pléiotrope). A l'inverse, un même gène peut faire l'objet de régulations multiples par plusieurs régulateurs. Ces interactions vont dessiner un réseau de régulation qui peut être complexe comprenant des sous-unités d'où peuvent émerger des propriétés fonctionnelles importantes, telles que des boucles de rétroaction ou au contraire des boucles d'amplification [114].

### 1.1.5 La diversité des ARN : au-delà de simples messagers

#### Les ARN non-codants

Jusqu'à maintenant, les ARN ont été présentés uniquement sous l'angle d'ARN messagers, en conformité avec le dogme central. Or, de nombreux travaux depuis les années 80 ont révélé l'existence d'ARN qui ont une activité dans la cellule sans pour autant avoir été traduits en protéines. Ces ARN sont connus sous le nom générique d'*ARN non-codants* (ARNnc).

L'implication de l'ARN dans différentes activités était déjà considérée comme très probable depuis la fin des années 60 [103, 145]. Mais c'est en 1982 que les travaux de Kruger et al. et de Guerrier-Takada et al. ont permis de mettre en évidence pour la première fois une activité catalytique de certains ARN [55, 80] (auto-épissage de l'ARN et l'activité de la ribonucléase P). Cette découverte leur a valu le prix Nobel de chimie en 1989. À la suite de ces découvertes, toute une classe d'ARN s'est dessinée : des ARN possédant une activité enzymatique, appelés ribozymes. Dans la plupart des cas, les ribozymes exploitent la capacité de l'ARN à se replier sur lui-même et à former des structures plus ou moins

complexes [33].

Certains de ces ARNnc sont impliqués dans des activités centrales. Par exemple, les ribosomes, acteurs principaux de la traduction, sont des complexes comprenant des ARN appelés ARN ribosomiques (ARNr) qui sont les ARN les plus abondants dans la cellule. De même, toujours durant la traduction, le lien entre les triplets de bases d'ARN et les acides aminés est réalisé grâce à des ARN dits de transfert (ARNt).

Les ARNnc vont pouvoir aussi réguler la transcription en modifiant localement la structure dans un chromosome de telle sorte qu'un gène soit accessible ou, à l'inverse, inaccessible à la machinerie de transcription. Ainsi, par exemple, deux ARNnc, Xist et Tsix, sont impliqués dans l'inactivation aléatoire d'un des deux chromosomes X chez les mammifères femelles. Xist va recouvrir un chromosome X et induire son inactivation alors Tsix va l'inhiber dans cette activité [92].

L'ensemble des ARNnc connus est regroupé dans une base de données appelée Rfam [50]. Cette base de données contient à ce jour (Rfam 12.0 : juillet 2014) plus de 19 623 515 séquences réparties en 2 450 familles [117], ce qui donne un aperçu quantitatif de la diversité considérable de ces éléments génétiques.

#### Les petits ARN interférents

Les quinze dernières années ont vu le développement d'un domaine de recherche considérable qui a révélé l'importance d'une classe d'ARN particulière caractérisée par une taille très réduite (21-24 nucléotides) et spécifiquement dédiée au *silencing* des gènes : les petits ARN *interférents* (siARN).

Ces petits ARN partagent un ensemble de mécanismes aussi bien dans leur synthèse que dans leur activité biologique, mais ils sont hétérogènes et comprennent plusieurs catégories, qui sont les *trans-acting siRNA* (ta-siARN), les *natural antisense transcript-derived siRNA* (nat-siARN), *long siRNA* (lsiRARN) et les microARN (miARN), qui vont nous intéresser particulièrement.

## 1.2 Les miARN

Les miARN sont de petits ARNnc, généralement d'un taille allant de 21 à 24 nucléotides, capables d'inhiber la traduction d'un ARNm en protéine, soit en bloquant la traduction de celui-ci, soit en le clivant menant à sa dégradation rapide [167]. Le mode d'action des miARN est caractérisé par une reconnaissance de leurs cibles par complémentarité de séquence et pose des questions intéressantes à la fois de biologie et d'informatique visant à décrire la structure de ces réseaux de régulation.

### 1.2.1 Historique

#### Découverte chez *Caenorhabditis elegans*

Les miARN ont été décrits pour la première fois par Lee, Feinbaum et Ambros (1993) [93] chez le nématode *Caenorhabditis elegans*. Ces travaux portaient sur un gène, *lin-4*, qui agit durant les stades larvaires, en affectant le déclenchement d'événements essentiels au développement de l'animal. Dans cette étude, les auteurs se sont rendu compte que ce gène ne codait pour aucune protéine, mais produisait deux petits ARN de taille 22 et 61 nucléotides. De plus, cet ARN contenait une séquence complémentaire à un élément répété retrouvé dans la région non traduite à l'extrémité 3' de l'ARNm (3'UTR), du gène *lin-14*. Des études précédentes avaient montré que *lin-4* était responsable de la régulation négative de *lin-14* et que cette régulation avait lieu à un moment situé entre la transcription et la traduction. Ces éléments ont permis de proposer un modèle selon lequel *lin-4* est capable d'inhiber la traduction de *lin-14* par une relation ARN-ARN complémentaires (voir également [175]).

Jusqu'en 2000, peu de recherches apportèrent des éléments nouveaux sur cette classe de petits ARN régulateurs. Un autre petit ARN similaire fut découvert chez *C. elegans*, *let-7* [132], un autre régulateur dans le développement larvaire de *C. elegans* en 2000, ce qui a introduit l'idée que ces éléments génétiques pourraient être non pas des exceptions mais constituer une classe de régulateurs à part entière. Sur cette base, les années 2000 ont connu une véritable explosion des études portant sur ces éléments génétiques.

Des homologues de *let-7* ont ensuite été identifiés dans une large variété d'espèces animales, invertébrées et vertébrées [123]. Aucun homologue de *let-7* n'a cependant été retrouvé chez les éponges et les cnidaires coté animal, chez *Escherichia coli* pour les bactéries et chez *Arabidopsis* pour les végétaux, montrant de ce fait que ce petit ARN n'est pas universel et est apparu le long d'une lignée spécifique. Enfin de nombreux nouveaux petits ARN furent ensuite rapidement découverts, chez *C. elegans* dans un premier temps [85]. La reconnaissance de propriétés communes à ces éléments génétiques a abouti à la formulation d'une classe particulière de petits ARN non codants désignée sous le nom de microARN, ou miARN, en référence à leur petite taille [90]. Depuis lors, de très nombreux nouveaux miARN ont été identifiés dans la quasi-totalité des espèces où ils ont été recherchés, et ils sont impliqués dans la plupart des processus cellulaires.

#### Un modèle similaire chez les plantes

Chez les plantes, la découverte des miARN a été consécutive à leur découverte et leur caractérisation chez les animaux. Des mécanismes de *silencing* post-transcriptionnel de gènes (PTGS) faisant intervenir des ARN complémentaires étaient déjà connus et impliquaient une protéine dont l'activité était identique à la protéine animal Dicer [141]. Chez les animaux, cette protéine intervient dans les mécanismes d'interférence par ARN [15], mais est aussi responsable de la découpe du miARN non mature (le précurseur) en une séquence de 21-24 nucléotides [51, 62], le miARN mature. Cette découverte d'une protéine similaire chez les plantes a mené plusieurs groupes à proposer simultanément que les plantes possèdent également des ARN qui partagent des caractéristiques similaires aux miARN des animaux [99, 122, 131]. Les travaux suivants ont ensuite rapidement montré qu'ils était largement impliqués dans la répression de l'expression des gènes contrôlant en particulier les processus du développement [68, 135].

### 1.2.2 Le mode de fonctionnement des miARN chez les plantes

#### Du gène au miARN mature

La synthèse de miARN nécessite plusieurs étapes de maturation. À l'inverse des animaux, où une partie importante des miARN est produite à partir de séquences introniques, chez les plantes la majorité des miARN est produite à partir de gènes qui leur sont spécifiquement dédiés (mais peuvent être tout de même aussi produits par des séquences introniques [128]). De façon générale, un gène ne résulte en la production que d'un seul miARN bien que dans de rares cas un même gène puisse être à l'origine de plusieurs précurseurs de miARN via la formation de plusieurs tige-boucle dans un même transcrit [156].

Tout comme pour les gènes qui sont traduits en protéines, c'est l'ARN polymérase II (Pol II) qui est chargée de la transcription des gènes de miARN (Figure 1.4). Ces gènes utilisent ainsi la même machinerie que le reste des gènes et peuvent donc être soumis aux mêmes mécanismes de régulation transcriptionnelle et s'intègrent au sein des réseaux de régulation de la cellule [111]. De la même façon que les ARNm sont modifiés, l'ARN produit par les gènes de miARN va subir un ensemble de modifications comme l'ajout d'une coiffe en 5', d'une queue poly-A en 3' [180] et parfois même un épissage [69].

Le transcrit ainsi obtenu est appelé *transcrit primaire*, ou pri-miARN. La taille de ce pri-miARN peut varier entre 50 et 900 nucléotides [17, 27]. Le pri-miARN forme des structures secondaires de taille et de forme variables. Il est cependant caractérisé par l'existence d'une tige imparfaite issue du repliement par complémentarité de séquence du pri-miARN [113]. Le pri-miARN va subir plusieurs modifications (Figure 1.4). Tout d'abord, un complexe enzymatique comprenant DCL1 va procéder à un premier clivage précis, généralement au niveau de la tige. À cette étape, l'intermédiaire est appelé *précurseur*, noté pré-miARN. Le pré-miARN hérite de la structure en tige-boucle du pri-miARN. Le pré-miARN va être clivé de nouveau par DCL1 pour n'obtenir plus qu'une structure double brin d'environ 20 à 24 nucléotides, un *duplex* composé du miARN et de sa séquence quasi-complémentaire, le miARN*, ou guide. Les imperfections de la tige au niveau du duplex (renflements et *mismatchs*) semblent être importantes dans la reconnaissance par DCL1. Cette structure va subir

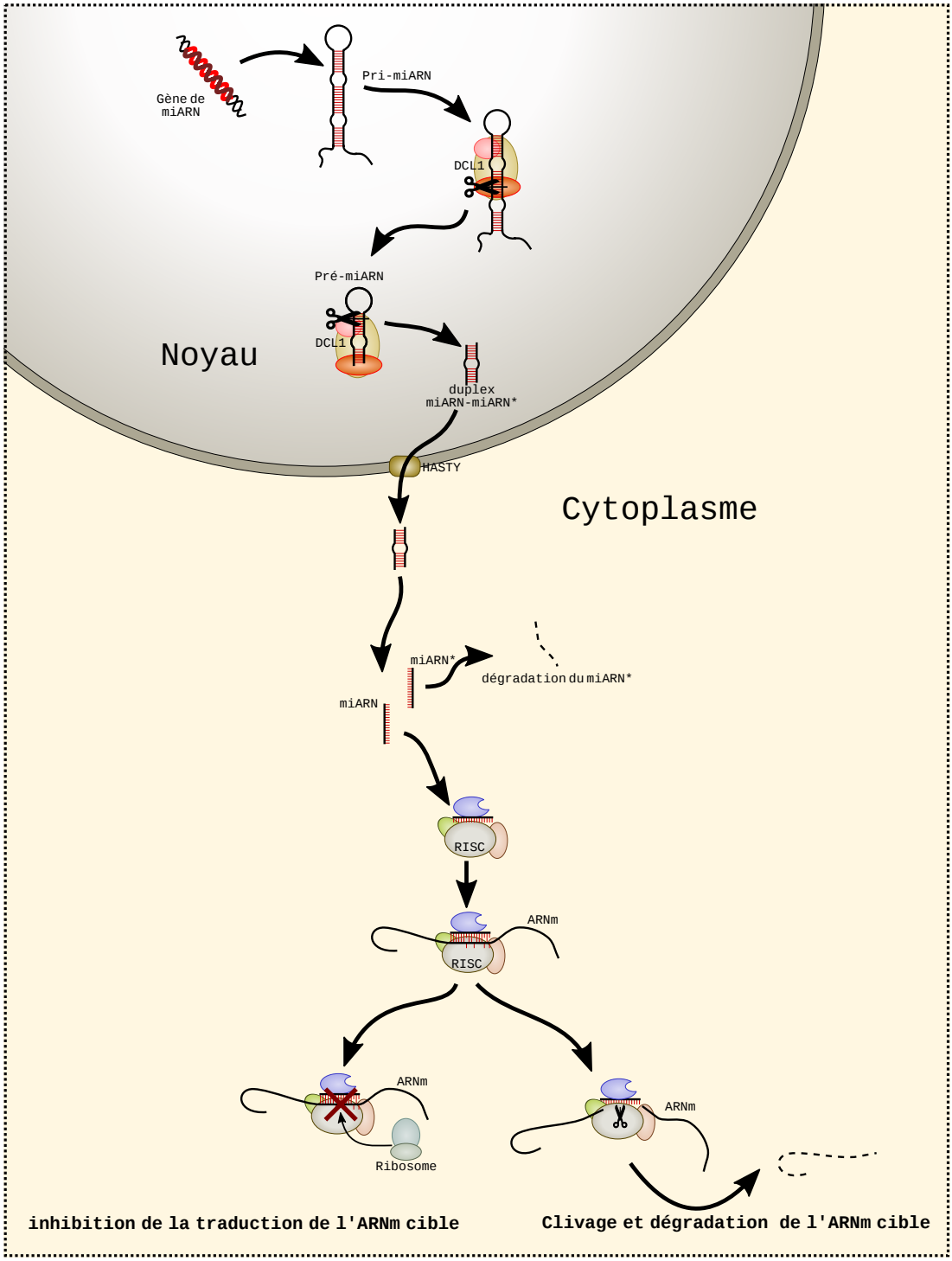



FIGURE 1.4 – Biogenèse des miARN et mode de *silencing* d'un ARNm cible dans le modèle végétal.

une méthylation par HEN1. Ce duplex est ensuite exporté du noyau vers le cytoplasme de la cellule par le transporteur HASTY.

Le duplex est alors dissocié par une hélicase en deux molécules d'ARN simple brin : le miARN mature d'un coté et miARN* de l'autre. La localisation de cette étape n'est pas très bien connue, et pourrait aussi bien être nucléaire que cytoplasmique. A ce stade, le miARN mature est associé à la protéine Argonaute (AGO) et forme le complexe RISC (*RNA-induced silencing complex*) qui sera le véritable effecteur du *silencing*. Le complexe RISC peut également inclure d'autres partenaires protéiques. Alors que chez l'humain, il existe huit protéines Argonaute, chez *A. thaliana* il en existe dix paralogues dont les fonctions sont associées à différents mécanismes de *silencing*. Les miARN sont majoritairement associés à AGO1.

#### Mécanismes de *silencing*

Les miARN, chargés dans un complexe RISC possèdent chez les plantes une forte complémentarité avec une partie de la séquence de leurs ARNm cibles. C'est grâce à cette complémentarité que le miARN guide spécifiquement le complexe RISC sur l'ARNm ciblé. Ce ciblage est la base de différents mécanismes d'inhibition de l'expression. Dans certains cas, la fixation du complexe RISC sur l'ARNm va empêcher la transcription en bloquant l'activité du ribosome. Chez les animaux, ce mécanisme est courant et a été décrit en détails, incluant le blocage de l'assemblage du ribosome sur l'ARNm, le détachement du ribosome alors qu'il est en cours de traduction, la dégradation de la protéine au fur et à mesure de la traduction et le recrutement d'autres protéines provoquant la déstabilisation de l'ARNm qui est alors dégradé [153]. Dans d'autre cas, le *silencing* s'effectue au travers du clivage de l'ARNm réalisé par la protéine AGO1 contenue dans le complexe RISC. Ce clivage est permis par l'activité RNAse du domaine PIWI de la protéine AGO1. Chez les animaux, le blocage de la traduction semble le plus fréquent, tandis que chez les plantes le clivage semble prédominant [10]. Dans la majorité des cas décrits à ce jour pour les plantes, le ciblage semble s'effectuer sur une portion de l'ARNm (régions exoniques ou UTR). Je réfèrerai à ce mode de ciblage comme le mode "canonique". Cependant, des études récentes

ont montré que d'autres modes de ciblage, encore mal connus, sont possibles. En particulier, [112] a montré que certaines cibles de miARN étaient localisées dans les introns de certains gènes chez le riz. Dans ce cas, il est envisageable que l'ARN pré-messager soit en fait la cible réelle malgré sa demi-vie limitée au sein de la cellule. Plus récemment encore, [159] a démontré expérimentalement l'existence de cibles de miARNs en amont du site d'initiation de la transcription du gène SCR (S locus cystein rich protein) au locus d'auto-incompatibilité chez Brassica. Dans ce cas, l'ARNm ne peut pas être la cible, et un mécanisme de méthylation de l'ADN lui-même semble être responsable du silencing du gène cible. Ces deux exemples illustrent que les modes de ciblage peuvent-être plus divers qu'initialement envisagés dans la littérature. Je réfèrerai collectivement à ces mécanismes comme les modes "non-canoniques".

## 1.3 Recherche de cibles de miARN

Nous venons de voir le mode d'action des miARN en tant que régulateurs post-transcriptionnel en ciblant l'ARNm. Un aspect central de la compréhension de la structure des réseaux de régulations qui en émerge consiste à connaître les cibles de chaque miARN du génome. Des méthodes expérimentales de recherche de cibles *ex nihilo* existent, mais cette tâche reste l'un des verrous principaux du domaine en raison de la difficulté de la prédiction et de la validation des cibles de miARN. Les approches expérimentales mises en œuvre peuvent se révéler particulièrement difficiles, longues et coûteuses. De ce fait, de nombreux travaux ont cherché à prédire *in silico* ces cibles pour guider et faciliter leur identification expérimentale.

Après avoir présenté quelques méthodes expérimentales pour l'identification de cibles, je présenterai les principales caractéristiques des cibles connues par rapport au miARN associé afin d'établir un modèle et présenter les principaux outils de prédiction *in silico* disponibles dans la littérature.

### 1.3.1 Approche expérimentale

#### Analyse différentielle

La façon classique de tester l'effet d'un régulateur consiste à observer la variation de l'expression des gènes en fonction de celle du régulateur supposé. La variation de l'expression du régulateur est réalisée par manipulations génétiques visant à sur-exprimer ou sous-exprimer le régulateur putatif [83].

L'analyse différentielle de l'expression du gène régulé peut être réalisée au niveau des protéines à l'aide d'une approche protéomique ou plus classiquement via une approche de mesure d'expression au niveau de l'ARNm par qPCR, puce à ADN ou plus récemment à l'aide du séquençage haut débit d'ARN (RNA-seq). Une limite majeure à cette approche différentielle consiste en la difficulté de distinguer les variations induites directement par le régulateur des variations indirectes via la perturbation d'autres réseaux de régulation.

**Analyse des interactions moléculaires**

Une autre manière de révéler l'action d'un régulateur consiste à observer directement les interactions moléculaires qu'il réalise avec ses cibles. Plusieurs techniques mettent à profit ces interactions, qui cherchent à concentrer les ARNm conjointement avec la molécule avec laquelle ils interagissent. Dans le cas des miARN, le *silencing* repose sur l'action du complexe RISC au niveau de l'ARNm. Une première approche consiste à marquer un miARN avec une molécule de biotine dont la forte affinité à la streptavidine permet de capturer non seulement le miARN, mais également le fragment d'ARNm avec lequel il interagit spécifiquement [119]. Une seconde approche permettant une description sans *a priori* consiste à immunoprécipiter le complexe RISC à l'aide d'un anticorps spécifique, puis à analyser le produit grâce à une puce à ADN ou par séquençage des fragments d'ARNm associés [70]. Une irradiation aux UV est généralement réalisée en amont pour renforcer les liaisons entre les protéines du complexe RISC et l'ARNm (méthode HITS-CLIP pour c*ross-*l*inking and* i*mmuno*p*recipitation with* hi*gh-*t*hroughput* s*equencing* )[30].

**Analyse des ARN clivés**

Dans les cas de *silencing* par clivage de l'ARNm, une façon puissante et directe d'identifier les ARNm ciblés est de détecter les produits du clivage. L'ARN clivé est spécifiquement séquencé à l'aide de méthodes comme PARE (*parallel analysis of RNA ends*). L'ensemble des séquences ainsi obtenues s'appelle le dégradome. L'analyse consiste à retrouver les gènes des ARN qui ont été clivés et par similarité de séquences au niveau du site de clivage retrouver le miARN qui peut être à l'origine de ce clivage. Cette technique a été utilisée avec succès sur *A. thaliana* [3, 47]. Un pipeline bioinformatique, dénommé CleaveLand, a été réalisé afin de détecter les ARNm clivés par un miARN [2].

Ces méthodes restent particulièrement difficiles à mettre en œuvre, longues, onéreuses et leur sensibilité peut être faible car dépendante du niveau d'expression du gène ciblé. Un gène donné spécifiquement exprimé dans un tissu va être par exemple plus difficilement identifié comme cible qu'un gène très fortement exprimé dans de nombreux tissus. Par ailleurs, le nombre important de cibles po-

tentielles et la diversité de mode de ciblage rend la validation expérimentale par analyse différentielle de toutes les cibles potentielles une tâche hors d'atteinte dans de nombreux cas.

La recherche de cibles *ex nihilo* peut cependant être soutenue par une prédiction *in silico* au préalable et permettre ainsi de diriger les méthodes *in vitro*.

### 1.3.2 Critères d'identification de cibles pour une approche *in silico*

#### Complémentarité presque parfaite

Très vite, il a été remarqué que chez les plantes, et contrairement aux animaux, la complémentarité entre le miARN et l'ARNm était presque parfaite [135]. Ainsi, la première méthode mise en œuvre pour trouver les cibles fut de les rechercher parmi les ARNm les séquences fortement complémentaires avec les miARN. Les premières études ont cherché à proposer un modèle qui maximise le nombre d'occurrences pour un miARN tout en gardant un bruit de fond faible. Ce bruit de fond correspond à des occurrences dues au hasard, et peut être mesuré avec une séquence aléatoire. C'est ainsi que Rhoades et al. a mesuré le nombre d'occurrences de séquences complémentaires aux miARN d'*A. thaliana* dans le génome, et l'a comparé au nombre d'occurrences obtenues avec des versions mélangées aléatoirement de ces même miARN. La conclusion est qu'un bon candidat devait contenir seulement 3 mismatchs sur 20 nucléotides successifs [135] (Figure 1.7).

Bien sûr, ce simple critère de complémentarité est une simplification, et plusieurs éléments permettent de raffiner la description de l'interaction. Parmi ces éléments, la présence de gaps a été documentée dans plusieurs interactions validées expérimentalement. Par exemple, l'appariement entre miR162 et l'ARNm de DCL1 possède un gap dans l'ARNm entre les nucléotides 7 et 8 du miARN relativement à son extrémité 5', sans que cela ne semble perturber l'interaction [179]. Par ailleurs, la simple complémentarité ne prend pas en compte le fait que les appariements G-U peuvent contribuer à stabiliser l'interaction, quoique de façon moins intense. Un tel appariement est appelé *wobble* (bancal). Ce type de

mismatchs doit donc être considéré comme de moindre importance. Conventionnellement, pour les miARN, les appariements G-U ne vont compter que pour un demi-*mismatch*. Ainsi miR319 (mir-JAW) est capable de s'apparier à plusieurs de ses cibles avec 4 ou 5 *mismatchs* et ce grâce à plusieurs appariements G-U [121].

**Calcul de l'énergie libre de liaison**

Afin que le miARN, incorporé dans le complexe RISC, reste attaché à l'ARNm, il est nécessaire que la liaison entre les deux ARN soit suffisamment forte. Plus l'énergie libre d'une liaison est basse, plus celle-ci est stable. La complémentarité entre les deux ARN est une simplification de cette énergie libre. L'importance des appariements G-U est ainsi minimisée dans le calcul du score d'appariement, car énergiquement ce *mismatch* est bien meilleur qu'un autre *mismatch*. Des modèles permettent aujourd'hui d'avoir une bonne estimation de cette énergie. L'énergie libre des appariements des miARN avec leurs cibles prédites a été donc étudiée. Bien que les couples expérimentalement validés se retrouvent généralement avec la meilleure énergie libre, ce critère ne permet pas de distinguer sans ambiguïté les couples fonctionnels des couples non fonctionnels [144]. L'énergie de liaison n'est donc pas le seul critère dans la reconnaissance d'une cible par le miARN et la plus-value apportée par la prise en compte de ce paramètre pour la prédiction des cibles n'est pas évidente.

**Sites d'interaction importants**

Une autre observation réalisée très tôt a montré qu'une substitution d'une seule base pouvait résulter en l'inactivation de l'inhibition de *PHB* et *PHV* par le miARN miR165 [109, 135]. Bien que dans le modèle animal l'importance de la complémentarité du miARN et sa cible soit moindre que chez les plantes, l'existence d'un "core élément" en région 5' de Lin-4 chez *C. elegans* a été notée très tôt [175]. D'autres observations ont montré qu'une bonne complémentarité de la cible avec la région à l'extrémité 5' du miARN est cruciale pour l'inhibition alors que celle à l'extrémité 3' l'est moins [32, 38, 94, 152]. Les analyses cinétiques de l'activité RNAse du complexe RISC ont démontré que l'appariement de cette région avec la cible contribue de façon importante à la liaison du complexe

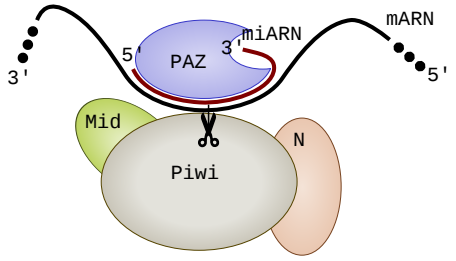
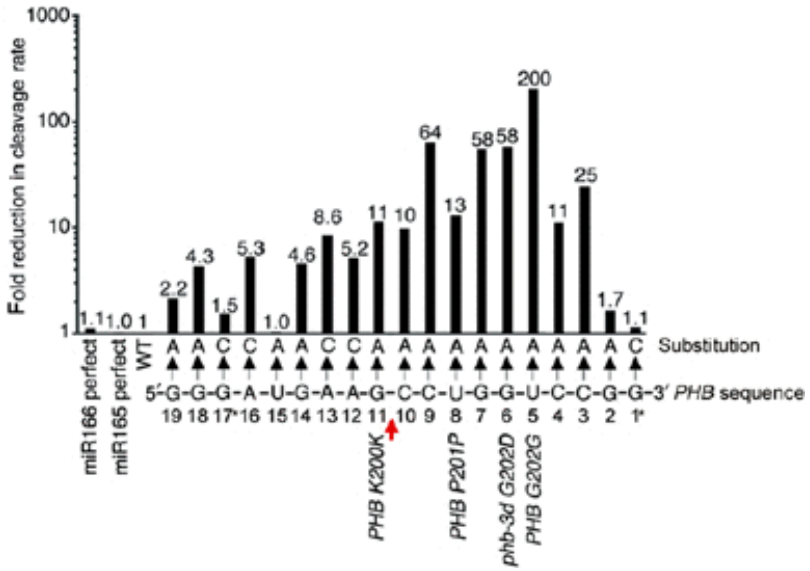



FIGURE 1.5 – Vue schématique de l'interaction du miARN avec sa cible dans le complexe RISC. La plus faible importance de l'extrémité 3' du miARN dans l'interaction avec sa cible serait expliqué par le fait que celle-ci interagirait initialement avec le sous-domaine PAZ et la région 5' du miARN.

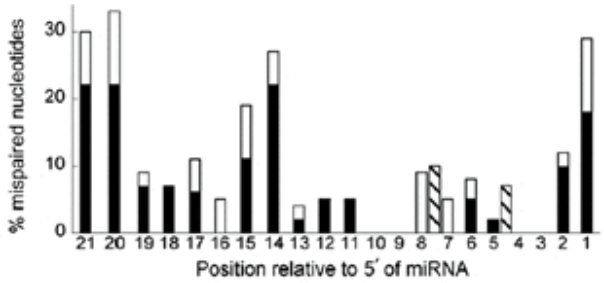
RISC à sa cible chez *Drosophila melanogaster* (drosophile) dans le cadre des siARN, des ARN non codants très similaires aux miARN [56]. De plus, ces mêmes travaux ont montré qu'une complémentarité parfaite aux positions 2-12 peut être suffisante pour observer un clivage *in vitro*. Une hypothèse avancée permettant d'expliquer l'importance de cette région 5' serait que le miARN se présente d'abord à sa cible via sa région 5' avant de s'apparier totalement sur toute sa longueur (Figure 1.5). Ainsi, une région faiblement complémentaire ne pourrait pas assurer une liaison assez forte pour que le complexe RISC puisse rester attaché suffisamment longtemps pour finir l'appariement total [56].

Afin de vérifier si cette observation était également valable chez les plantes, des expériences de mutations sur un miARN ont été réalisées afin de voir comment le taux de clivage d'un ARNm ciblé varie en fonction de la position à laquelle la mutation était introduite. Il a été observé que des substitutions sur la région centrale et 5' du miARN font baisser le taux de clivage, alors que des substitutions dans la région 3' du miARN ont une influence moindre (Figure 1.6a). De plus, l'observation des appariements entre les miARN et leurs cibles expérimentalement validées montre aussi que la région centrale et 5' du miARN est moins sujette aux erreurs, les gap et les *mismatch* sont plus rares aux positions allant de 3 à 10 en particulier les segments 3-4 et 7-10 qui ne possèdent aucun *mismatch* et les segments 3-4 et 9-10 qui sont observés toujours parfaitement appariés (Figure 1.6b) [107].

Les résultats de l'analyse des produits de dégradation montrent que le com-




(a)




(b)

FIGURE 1.6 – Importance des sites d'interaction : a Réduction du taux de clivage en fonction de la position d'une substitution simple [107] : une substitution simple sur la cible (gène *PHB*) n'a pas le même effet sur la réduction du taux de clivage en fonction de sa position avec une forte réduction de celle-ci si la substitution se trouve sur la région 3' de la cible et 5' du miARN (source : [107]). b Compilation des fréquences des erreurs d'appariement entre le miARN et sa cible en fonction de la position chez *Arabidopsis* [107] : *Mismatch* (barre pleine), G :U *wobble* (barre blanche) et *gap* (barre hachurée). La fréquences des erreurs d'appariement montre de même que la région 3' de la cible et la région 5' du miARN ont peu d'erreurs (source : [107])..

plexe RISC clive généralement l'ARNm ciblé entre le 10e et le 11e nucléotides du miARN. L'importance de ce site a tout d'abord été observée pour les siARN de la drosophile [37] puis pour les miARN chez *A. thaliana* [98]. Une hypothèse proposée est que AGO1 a besoin d'un appariement parfait à ce niveau pour pouvoir couper l'ARN. L'analyse expérimentale de la tolérance de chaque position au *mismatch* (Figure 1.6a) montre cependant que ces deux positions agissent effectivement sur l'efficacité du miARN mais ne semblent pas se distinguer réellement des autres positions de la région centrale et 5' [107]. Malgré cela, il a été remarqué ailleurs que plusieurs fausses prédictions de cibles possédaient la même énergie libre qu'une cible réelle mais avaient un *mismatch* aux positions 10 ou 11 [144]. Néanmoins un appariement imparfait à cette position empêche le clivage de l'ARNm par le complexe RISC [19].

#### Accessibilité de la cible

Au-delà de l'énergie libre globale de liaison entre le miARN et sa cible, certains auteurs ont proposé de s'intéresser à l'accessibilité de la zone cible. En effet, l'ARNm étant simple brin, il peut se replier sur lui-même (Figure 1.2b). Ce mécanisme est par exemple à la base des riboswitchs, un système de régulation de la traduction dans lequel la conformation prise par l'ARNm va bloquer ou activer la traduction [162]. Dans le cadre des miARN, si la zone potentiellement ciblée par le miARN est déjà appariée à un autre segment de l'ARNm dans un appariement fort, alors cette zone ne serait plus accessible au miARN qui serait donc incapable de s'y apparier. Chez les animaux, cette accessibilité se révèle être un critère bien plus important dans l'action du miARN qu'un appariement solide [58]. Chez les plantes, il a été mis en évidence qu'une structure secondaire solide présente dans un ARN viral permet à celui-ci d'échapper au clivage par RISC dirigé par un petit ARN [65]. Cependant, dans le cadre de miARN, il n'existe à ma connaissance aucune étude démontrant à ce jour que l'accessibilité de la cible soit primordiale dans le modèle végétal.

### Conservation phylogénétique des cibles

Outre les propriétés intrinsèques des cibles, certains auteurs ont proposé d'utiliser la conservation phylogénétique comme un critère additionnel appuyant la validité d'une cible [131, 135]. La conservation d'une séquence entre deux génomes d'espèces différentes est le signe d'une sélection purifiante au niveau de cette séquence [158]. Lorsqu'un miARN est conservé entre deux espèces phylogénétiquement éloignées, prédire des cibles de ce miARN sur le même gène est un bon signe quant à la validité de la prédiction. Ainsi, dès les premières études sur la prédiction de cibles chez les plantes, ce critère fut utilisé pour appuyer le modèle proposé en comparant les cibles de miARN chez *A. thaliana* et chez le riz (*Oryza sativa*) [131]. Cependant, ce ne peut être un critère éliminatoire. En effet, il existe des miARN qui sont apparus très récemment et ne sont partagés qu'au sein de groupes phylogénétiques réduits. L'hypothèse serait que ces miARN soient apparus plus récemment. Ainsi, il a été fait une distinction entre des miARN dit conservés, lorsqu'ils sont retrouvés chez des espèces éloignées, et moins conservés, lorsqu'ils sont retrouvés uniquement chez une seule espèce ou entre des espèces très proches. La distinction exacte entre les deux varie selon les auteurs [9].

De la même façon, il est possible que la séquence des miARN et de leurs cibles ait varié rapidement et ne soit plus reconnaissable entre espèces, tout en ayant conservé une interaction fonctionnelle à chaque étape de leur évolution. Dans ce cas, la conservation phylogénétique ne sera pas un bon indice de leur rôle fonctionnel.

### Un modèle encore à définir

Comprendre le mode d'interaction entre le miARN et sa cible et en déduire des règles n'est pas aisé, et des observations peuvent se révéler contradictoires. Ainsi par exemple, la découverte d'un renflement de 6 nucléotides en position 6-7 du miARN miR398 lorsqu'il est apparié avec le gène At5G20230 [20] ne s'accorde pas avec le critère de forte complémentarité entre le miARN et sa cible.

A ce stade, l'enjeu des outils de prédictions de cibles est donc de parvenir à identifier grâce à des critère simples et rapides un nombre de cibles aussi

| | | Targetfinder | Tapirfasta | Tapirhybrid | Target-align | PsRNATarget | P-TAREF |
|---|---|---|---|---|---|---|---|
| | outil de pré-filtrage | FASTA | FASTA | RNAHybrid | Smith-waterman | Ssearch | RNAhybrid |
| | score | Pos. dép. | Pos. dép. | Pos. dép. | Global | Pos. dép. | |
| | Limite | 4 | 4 | 4 | 4 | 3 | |
| | Gestion de la *seed* | Score*2 | Score*2 | Score*2 | 1 *mismatch* max | Score+0,5 | |
| Conditions | G :U limite | | | | 6 | | *Machine learning* |
| | *indel* limitation | 1 | | | 4 | | |
| | *mismatch* consécutif | | | | 2 | | |
| | MFE ratio | | 0,7 | 0,7 | | | |

Tableau 1.1 – Résumé de quelques outils de recherche de cibles. Targetfinder, Tapir, Target-align et psRNATarget sont quatre outils de recherche de cibles de miARN chez les plantes maintenus et bien cités. Les algorithmes sont assez similaires. Ainsi, la recherche de cibles s'effectue tout d'abord à l'aide d'un algorithme de recherche local. Cet algorithme peut être un algorithme d'alignement exact comme pour Target-align ou psRNATarget ou heuristique comme pour Targetfinder ou Tapir avec le moteur FASTA. Il peut être aussi basé sur l'énergie d'hybridation ARN-ARN comme le fait Tapir avec le moteur RNAHybrid. Les cibles potentielles sont ensuite validées à l'aide d'un ensemble de critères. Le critère principal est le score d'alignement.

important que possible, sachant que de nombreuses exceptions sont possibles, et que les travaux de biologie continuent à décrire des interactions non canoniques. A terme, l'ambition est que cette interaction entre prédiction théorique et validation expérimentale permette de faire progresser notre compréhension des réseaux de régulation.

### 1.3.3 Les outils de prédiction de cibles

La recherche de cibles reste donc à ce jour un défi méthodologique important et les principaux outils informatiques ont cherché à exploiter différentes propriétés des cibles expérimentalement validées. Nous allons voir les principaux outils existants utilisés pour la recherche de cibles de miARN.

#### Recherche par complémentarité dans les ARNm, premier filtrage des cibles

Au vu du fonctionnement des miARN ciblant les ARNm, les études vont généralement limiter la recherche de cibles uniquement aux régions transcrites du génome et non au génome complet [135]. Cette limitation permet de réduire grandement la recherche, les régions codantes représentant entre 20% environ chez *A. thaliana* [7] et à peine 8% du maïs (*Z. mays*). Alors que dans le modèle

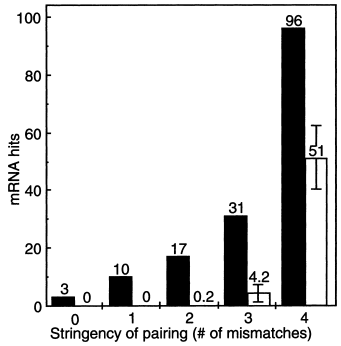



FIGURE 1.7 – Nombre d'occurrences de cibles potentielle en fonction du nombre d'erreurs autorisées. En barre pleine 16 miARN connu en barre blanche moyenne pour 10 cohortes de faux miARN générées aléatoirement à partir de la séquence mélangé des 16 miARN (source [135]).

animal, les cibles sont principalement retrouvées dans les région 3'UTR [86], dans le modèle végétal, aucune observation similaire n'a été faite et les cibles peuvent être présentes à toutes les positions des transcrits [135]. Cependant, contrairement au modèle animal, les miARN des végétaux sont fortement complémentaires à leur cible. Cela constitue donc le critère principal de la recherche des cibles et les premières études publiées utilisaient comme unique critère la complémentarité [135]. Le nombre de mismatchs est utilisé comme score en limitant leur nombre à seulement 3 sur une fenêtre de 20 nucléotides. En effet, ceci a permis d'avoir un nombre plus important d'occurrences tout en ayant une faible chance d'être dû au hasard (Figure 1.7).

Cette première étude et plusieurs études suivantes [68, 128, 135] ont utilisé PatScan [35] pour effectuer la recherche de cibles. Cet outil générique permet en effet de rechercher des motifs dans une séquence biologique au travers d'une expression régulière. Il a permis aussi de tester de nouveaux modèles et permettre les *gaps*. Cependant, PatScan n'est pas un outil destiné spécifiquement à la recherche de cibles de miARN : il ne possède pas la souplesse nécessaire pour écrire une expression régulière décrivant des règles de complémentarité plus complexes nécessaires pour la recherche de cibles. Il ne permet pas facilement de mettre en place des notions comme une pénalité de score variable selon le type d'erreur (*gap*, *mismatch*, *wobble*) et sa position [4, 144].

Rechercher directement une région répondant à certains critères peut être particulièrement difficile [23]. Néanmoins, la recherche de similarité est un domaine de recherche essentiel en bioinformatique. C'est donc assez naturellement que des outils spécifiques à la recherche de cibles de miARN ont été développés en proposant généralement une recherche en deux étapes. La première étape, le pré-filtrage, consiste à rechercher rapidement à l'aide d'algorithmes efficaces des régions d'intérêt par leur forte complémentarité avec le miARN. La seconde étape consiste ensuite à tester un ensemble de critères définissant le modèle de cibles de miARN utilisé.

Pour la première étape, parmi les outils existants, il y a deux approches : soit utiliser des algorithmes exacts, soit utiliser des algorithmes heuristiques. Une recherche exacte va retourner tous les résultats attendus, mais ceci dans un temps d'exécution qui peut être long. Une recherche heuristique consiste à accélérer l'exécution en acceptant de ne pas être exhaustif. Un tel algorithme ne va donc pas forcément retourner tous les résultats attendus.

Parmi les algorithmes exacts utilisés, l'algorithme de Smith et Waterman [149] est le plus courant. Il retourne un alignement local de score optimal. Cependant, cet algorithme est particulièrement lent, même si des implémentations comme ssearch [125] offrent de nombreuses améliorations [48]. Les possibilités d'utiliser l'algorithme de Smith et Waterman à l'échelle du génome restent limitées en terme de temps d'exécution. Nous reparlerons de cet aspect dans le chapitre 2.

Parmi les algorithmes heuristiques, BLASTN [6] ou FASTA [124] sont les plus utilisés. Cependant, ces outils laissent passer des candidats. Ceci est particulièrement vrai pour BLASTN qui ne se montre pas adapté à la recherche de petites séquences, comme les miARN. Ainsi en recherchant les séquences des 218 miARN provenant de la base de données mirbase [49] avec 4 *mismatchs* sur 39 640 ADNc (séquence d'ADN complémentaire de l'ARNm mature) extraits de TAIR9 [134], BLASTN identifie uniquement 231 cibles pour 120 miARN, alors que ssearch trouve 863 cibles pour 207 miARN [29].

Enfin, une dernière possibilité pour sélectionner les régions de cibles potentielles est l'utilisation d'un algorithme de calcul d'énergie d'hybridation de l'ARN. De tels algorithmes sont cependant plus lents encore que les algorithmes

exacts de recherche par similarité. Cette approche est par exemple utilisée dans TAPIR [18], qui est un outil de recherche de cibles de miARN et permet de choisir entre FASTA et RNAHybrid [130] pour sélectionner les régions de cibles potentielles. La recherche de cibles avec RNAHybrid mène à un taux de vrais positifs équivalent à une recherche avec FASTA, mais avec un taux de faux positifs plus élevé. Ce résultat serait dû à une détection plus sensible des duplex plus faibles, ce qui lui permettrait de récupérer un plus grand nombre de cibles au prix d'un nombre de faux positifs élevé [18]. Cela illustre le fait que l'énergie d'hybridation ne semble pas apporter beaucoup dans la recherche de cibles par rapport à une recherche par simple complémentarité. Enfin, comme prévu, TAPIR, avec RNAHybrid, est plus lent à l'exécution d'un facteur 240 [151].

#### Critères sur les séquences miARN et cibles : second filtrage

Une fois les régions d'intérêt retrouvées par similarité, la seconde partie de la recherche va consister à tester un ensemble de critères faisant référence au différentes propriétés précédemment décrites. Les critères tournent principalement autour de conditions d'appariement entre le miARN et sa cible. Nous avons vu précédemment que plusieurs propriétés dans l'appariement entre le miARN et sa cible ont été observées, cependant les différents outils ne vont pas toutes les utiliser ou ne vont pas les implémenter de la même manière.

Le premier critère qui est couramment retrouvé dans les outils de recherche de cibles est un score basé sur des paramètres pour l'appariement pénalisant *mismatchs* et *gaps*. Les outils actuels accordent généralement une pénalité moindre à un appariement de type G-U, par rapport à un autre type de mismatch. Les pénalités généralement retrouvées sont 1 pour les *mismatchs* et les *gaps*, et 0,5 pour les G-U. Ce n'est pas une généralité. Dans PS-RNATarget [28] un *gap* va être plus fortement pénalisé avec une pénalité de 2. Si le score de l'alignement dépasse un certain seuil, la région candidate est éliminée.

Afin de prendre en compte l'importance de la région 5' du miARN dans l'appariement, la plupart des outils vont utiliser une notion de graine. Dans beaucoup d'outils, les erreurs sont plus fortement pénalisées dans cette région, tel que c'est proposé par Allen et al. [4]. Dans cette étude, les pénalités sont

doublées entre les nucléotides 2 et 13 et le seuil du score est fixé à 4. Cependant, dans d'autres outils, l'importance de cette graine ne se fait plus au niveau du score, mais en ajoutant un nouveau critère qui est de limiter le nombre et le type d'erreurs dans cette région, comme proposé par Schwab et al. [144]. Dans cette étude, il est proposé de n'autoriser qu'un seul *mismatch* entre les nucléotides 2 et 12.

En plus du score et de la graine, d'autres critères supplémentaires peuvent être utilisés. Le plus fréquemment retrouvé est un ratio de l'énergie libre de liaison entre le miARN et sa cible, par celle du miARN et de son complémentaire parfait comme le proposent Allen et al. [4] et Schwab et al. [144]. Ce critère est mis en œuvre dans les logiciels TAPIR, SoMART [95] ou Target_Prediction [155]. Par défaut, les outils proposent un ratio minimal d'environ 0,7. Le MFE peut être calculé à l'aide d'outils fournis dans le vienna RNA Package [60, 100]. Cependant, une façon simple et rapide d'approximer ce ratio est de compter le nombre de liaisons hydrogène. Un appariement C-G compte pour 3 liaisons alors qu'un appariement A-U compte pour 2, G-U pour 1, tout le reste comptant pour 0. C'est ce qui est fait dans SoMART [95].

Certains outils vont aussi interdire toute erreur aux positions 10 et 11 du miARN, tel que Schwab et al. l'avait proposé en raison de l'importance fonctionnelle de ces deux positions lors du clivage (Figure 1.5) [19]. Toutefois ce critère n'est pas toujours vérifié dans les cibles connues (Figure 1.6a et Figure 1.6b) [107] et, dans d'autre outils, ce critère n'est pas éliminatoire. Le cas échéant, il est utilisé pour indiquer que l'action du miARN sur cette cible ne se fait pas par clivage comme dans PsRNATarget [28]. Parmi les autres critères parfois retrouvés dans les outils, le nombre de *mismatchs* consécutifs est limité à 2, tel que proposé dans Schwab et al. [144] et dans Target-align [178]. Toujours dans les critères d'alignement, quelques outils vont affiner les limites d'erreurs imposées par le score en limitant en plus le nombre de certains types d'erreurs tels que les *gaps* et les appariements G-U dans l'appariement global. Targetfinder et miRU [182] vont ainsi limiter le nombre de *gaps* sur l'ensemble de l'alignement.

Enfin, l'outil p-TAREF [66] met en œuvre une approche originale pour définir un modèle de cibles. Il utilise une méthode d'apprentissage automatique sur un jeu de données de couples miARN-cible pour extraire ce modèle. Pour chaque

couple de ce jeu, il extrait la répartition des erreurs le long de l'appariement, c'est-à-dire la position des *mismatchs*, des appariements G-U et *gaps*. Pour valider un appariement entre un miARN et une cible potentielle, il recherche si celui-ci ne présente pas une répartition des erreurs similaire à un appariement qu'il a rencontré dans le jeu de données.

Enfin, en plus des conditions d'appariement, un dernier critère parfois utilisé est l'accessibilité de la cible. Ainsi, psRNATarget va utiliser RNAup du Vienna RNA package [60, 100] pour calculer l'énergie nécessaire pour défaire les structures secondaires autour du site de la cible sur l'ARNm. P-TAREF, quant à lui, va utiliser un apprentissage supervisé de la variation de la densité dinucléotidique des régions flanquantes, comme cela a été proposé par Heikham et Shankar [59]. Pour ce faire, il utilise une approche d'apprentissage cette fois sur un "SVR" (S*upport* V*ector* R*egression*) à l'aide de SVMTorch [24] pour extraire, d'une base de données de couples miARN-cibles fournie en entrée, des informations sur la composition en dinucléotides d'une fenêtre de 20 bases sur une région de 75 bases autour de la cible.

#### Comparaison de la performance des outils de prédiction de cibles

L'évaluation de la performance des outils de prédiction de cibles repose généralement sur la confrontation entre une base de données de cibles validées expérimentalement et les sorties d'outils de prédiction bio-informatique. La spécificité et la sensibilité sont des mesures statistiques permettant d'évaluer la validité d'une hypothèse, ici représentée par l'algorithme de prédiction de cibles. Un outil avec une forte sensibilité (appelée aussi rappel) est capable de retourner pratiquement toutes les cibles dans ses résultats. Un outil avec une forte spécificité va faire peu d'erreur dans les cibles retournées et les résultats obtenus sont donc plutôt fiables. Un bon outil est donc à la fois sensible, pour retrouver le maximum de cibles, mais aussi spécifique, pour avoir peu de fausses prédictions dans les résultats. Une prédiction d'un logiciel est appelée *vrai positif* (VP) si elle correspond à une cible déjà connue. D'autre part lorsque l'outil ne trouve pas une cible qui n'est effectivement pas connue, on appelle cela *vrai négatif* (VN). Un outil peut cependant prédire qu'il y a une cible dans une

région alors que ce n'est pas le cas. Cette prédiction est appelée *faux positif* (FP). Enfin, une cible peut ne pas être retrouvée par l'outil. C'est un *faux négatif* (FN). La sensibilité est donnée par $\frac{VP}{VP+FN}$ et la spécificité est donnée quant-à elle par $\frac{VN}{VN+FP}$ mais est aussi équivalant à $1 - \frac{FP}{FP+VN}$. Au lieu de la spécificité, il est parfois utilisé la précision donnée par $\frac{VP}{VP+FP}$. Cependant pour calculer ces valeurs, ils est nécessaire de bien distinguer toutes les classes. Or il n'est pas possible d'affirmer qu'un couple miARN-cible prédit par un outil mais absent du jeu de données ne soit pas une véritable cible, car les approches expérimentales ne sont pas exhaustives. De même, nous ne sommes pas en mesure de connaître les véritables couples qu'un outil n'a pas pu prédire si ceux-ci n'ont pas été non plus retrouvés par approche expérimentale.

Pour résoudre ce problème, si l'outil de prédiction est spécifique à des couples miARN-cible, il devrait produire un nombre plus faible de prédiction de cibles avec un faux miARN (Figure 1.7).

Cependant, l'ensemble des couples de miARN-cible connus constituera notre jeu de test et sera considéré comme un échantillon représentatif de l'ensemble des couples miARN-cible. Sous cette hypothèse, les valeurs statistiques décrites ci-dessus refléteront la qualité de la prédiction. Ainsi, comme le jeu de test est considéré comme représentatif, un outil avec une sensibilité élevée va potentiellement retrouver l'intégralité des cibles. De même, on s'attend à ce qu'un outil ayant une précision élevée retrouve tout de même un taux plus faible de cibles non incluses dans notre jeu de données qu'un outil avec une précision moins élevée retournant plus de faux positifs.

Il existe plusieurs travaux de comparaison d'outils de recherche de cibles de miARN chez les plantes [29, 31, 151]. Srivastava et al. [151] en particulier a réalisé une étude comparative récente des outils sur leur capacité à effectuer une recherche de cibles *de novo* sur un génome complet de différentes plantes (*A. lyrata*, *Glycine max*, *Solanum lycopersicum*, *Brassica rapa*, *Panax ginseng*, *O. sativa*, *Vitis vinifera*, *Triticum aestivum*, *Medicago truncatula*). Le premier point à noter est que sur les 18 outils sélectionnés, seuls 11 ont pu être utilisés. En effet, une partie d'entre eux n'est tout simplement plus disponible, une autre partie n'est pas adaptée à la recherche *de novo* sur un génome complet. En effet, certains outils sont destinés à la construction de miARN artificiels alors que SoMART a

besoin du dégradome pour cartographier les cibles de miARN.

Srivastava et al. ont réuni un jeu de couples miARN-cible d'espèces différentes expérimentalement validés de différentes sources (miBase V18, Arabidopsis small RNA project et littérature). Ils ont ensuite utilisé les différents outils pour confronter pour chaque espèce leurs miARN au transcriptome provenant de Phytozome V8.0. L'étude s'est d'abord intéressée à *A. thaliana* uniquement, puis a comparé ces résultats à ceux obtenus avec les autres espèces végétales.

Sur *A. thaliana*, p-TAREF et target-align se sont révélés être très lents, pour analyser l'intégralité des données avec un temps d'exécution de plus de 2 semaines. Du fait d'une trop grande différence dans les mécanismes de reconnaissance de cibles par rapport au modèle végétal, l'utilisation d'outils de prédiction pour un modèle animal n'était pas pertinente. En effet, miRanda [67], RNAhybrid [79] et TargetScan [94] ont prédit plus de 4000 cibles par miARN alors que le jeu de couples de miARN-cibles n'était que de 330 interactions pour l'ensemble des miARN.

Du point de vue de la qualité des résultats, Targetfinder [41] semble être l'outil le plus approprié pour la recherche de cibles de miARN chez *A. thaliana* offrant le meilleur score de précision et de rappel de 89% et 97% respectivement (Figure 1.8a). Au contraire, p-TAREF est, quant à lui, l'outil qui a le plus grand nombre de cibles prédites qui ne soient pas validées, résultant en une précision de seulement 2%. Le fonctionnement de cet outil est assez atypique, ne reposant pas sur un modèle fixé au préalable, mais sur de l'apprentissage automatique.

Un point important soulevé par cette étude est la différence de qualité des résultats dès lors que l'analyse n'est plus réalisée sur le génome d'*A. thaliana*. En effet, le rappel tombe à 43% lorsque les outils sont exécutés avec les paramètres qui ont donné les meilleurs résultats pour *A. thaliana*. Après optimisation de ces paramètres, afin d'obtenir les meilleurs valeurs de précision et de rappel, TargetFinder arrive à obtenir une précision de 70% et un rappel de 69% et psRNATarget une précision de 74% avec un rappel de 62% (Figure 1.8b). Cela semble montrer que les modèles utilisés par les cibles peuvent différer d'une espèce à l'autre, les outils étant généralement optimisés autour des interactions miARN-cible des espèces modèles.

Cela semble se confirmer lors de l'analyse des résultats qui suggère que pour


FIGURE 1.8 – Performance, redondance et complémentarité des outils de recherche de cibles, évaluées par la comparaison des prédictions de «vrais positifs» et de «faux positifs» lorsque les différents outils sont utilisés soit indépendamment soit en combinaison. Les prédictions ont été réalisées sur le transcriptome d'*A. thaliana* et comparées à 330 interactions miARN :ARNm expérimentalement validées et considérées comme «vrais positifs» (a) et sur le transcriptome d'autres espèces (*G. max*,*S. lycopersicum*, *B. rapa*, *P. ginseng*, *O. sativa*, *V. vinifera*, *T. aestivum*, *M. truncatula*) et comparé à 115 interactions miARN :ARNm expérimentalement validées (b). Deux types de combinaisons ont été comparées : soit l'union des résultats des deux outils soit leur intersection. Les paramètres des outils (et plus exactement l'option de limite de score) ont été optimisés de telle sorte que la différence entre les valeurs de rappel et de précision soient la plus petite possible lorsqu'ils sont utilisés seuls chez *A. thaliana* ou chez les autres espèces. (source :[151] )

un sous-ensemble de miARN non-retrouvés dans *A. thaliana* d'autres mécanismes sont en jeu dans la reconnaissance de la cible par le miARN. En effet, en étudiant plusieurs paramètres comme la longueur du premier appariement exact, du plus grand appariement exact, et du ratio du nombre de match sur le nombre de *mismatch*, ces éléments semblent statistiquement différents chez les espèces autres que *A. thaliana*. Ceci suggère que le modèle utilisé par les outils, bien qu'il fonctionne assez bien chez *A. thaliana*, passe à côté d'un ou plusieurs sous-ensembles de miARN absents chez *A. thaliana* mais présent chez les autres espèces.

Enfin, l'étude montre que combiner le résultat des outils est une solution possible pour améliorer la précision et le rappel des résultats, mais seulement dans le cadre d'espèces différentes d'*A. thaliana*. En effet, chez *A. thaliana*, l'union ou l'intersection de plusieurs outils a un effet très limité sur les résultats (Figure 1.8). Globalement Srivastava et al. [151] révèle donc une très grande hétérogénéité dans la performance des différents outils.

### 1.3.4 L'actuel défi de la recherche de cibles de miARN

Les nouvelles technologies de séquençage sont capables de produire d'importantes quantités de données. Les procédures de recherche de miARN et de leurs cibles *in vivo* se sont énormément développées et nécessitent des outils d'analyse de données adaptés. De plus, des études récentes [36, 112, 159] tendent à prouver l'existence de cibles de miARN dans des régions non codantes du génome. Ceci implique de travailler sur l'intégralité du génome et non plus seulement sur les seules régions transcrites.

En parallèle, le coût de séquençage a largement diminué, produisant toujours plus de séquences de génomes d'espèces non-modèles, sur lesquelles la recherche de cibles de miARN est appelée à se développer. Ainsi, il est observé une augmentation de la quantité de données à analyser aussi bien avec de nouveaux miARN que de séquences génomiques sur lesquelles chercher les cibles.

Il est donc nécessaire de proposer des outils capables de fournir un débit important et d'analyser rapidement ces données.

Nous avons pu voir que la phase préliminaire de recherche de séquences

complémentaires est cruciale dans la recherche de cibles de miARN. Or, il n'existe pas beaucoup d'outils capables de rechercher exhaustivement les occurrences d'une petite séquence (21 à 24 nucléotides) avec un taux d'erreurs élevé incluant des *gaps* dans une séquence de plusieurs millions de bases, telle qu'un génome, tout en étant rapide. Cela représente un important défi algorithmique, et c'est l'objet du travail de ma thèse.

Chapitre 2

# Recherche de motifs approchée, les graines 01*0

Dans le premier chapitre de ce document, nous avons présenté la problématique de la recherche de cibles de miARN dans les génomes de plantes. Nous avons montré que cette problématique débouchait sur un problème algorithmique bien défini : la recherche de motifs approchée, qui fait l'objet de ce deuxième chapitre.

Dans une première section, nous donnons les définitions nécessaires à la formalisation du problème. En section 2, nous présentons les approches classiques par programmation dynamique et les optimisations à base de graines. En section 3, nous expliquons comment ces approches peuvent être articulées avec une indexation des génomes. Enfin, en section 4, nous présentons notre contribution avec les graines 01*0 qui permettent de rechercher de courts motifs avec un taux d'erreurs élevé dans de longs textes, et dont la forme se prête à une recherche dans les structures d'index les plus sophistiquées.

## 2.1 Notations et Définitions

Nous introduisons les définitions de base dont nous avons besoin pour manipuler des mots.

Un *alphabet* est un ensemble fini dont les éléments sont appelés *lettres*. Dans ce document, il est noté $\Sigma$. La taille de cet alphabet, c'est-à-dire le nombre de lettres composant l'alphabet, est notée $\sigma$.

- Un *mot* $U$ est une suite finie de lettres prises dans l'alphabet $\Sigma$. Un mot peut n'être composé d'aucune lettre : c'est le *mot vide* et il est noté $\epsilon$.
- L'ensemble des mots sur l'alphabet $\Sigma$ se note $\Sigma^*$.
- La longueur d'un mot $U$ est le nombre de lettres de ce mot et s'écrit $|U|$.
- La *concaténation* de deux mots $U$ et $V$, notée $UV$, mais aussi parfois $U \cdot V$ pour éviter les confusions, est le mot de longueur $|U|+|V|$ dans $\Sigma^*$ composé des lettres de $U$ suivies des lettres de $V$.
- L'*indice* est une coordonnée caractérisant la position d'une lettre dans un mot. Par convention, la première lettre du mot $U$, $U \neq \epsilon$, a l'indice 0 et sa dernière lettre a donc l'indice $|U|-1$. Pour tout entier $i$ compris entre 0 et $|U|-1$, nous noterons $U_{[i]}$ la lettre d'indice $i$ de $U$.

**Exemple 1.** *L'ADN est un alphabet de taille* 4 *et l'ensemble des lettres qui le composent est* $\{A,C,G,T\}$. *CTAGTTAG est un mot de longueur* 8. *Les indices des différentes lettres sont* $\overset{0\,1\,2\,3\,4\,5\,6\,7}{CTAGTTAG}$.

On considère maintenant deux mots $U$ et $V$ de $\Sigma^*$.

— Un mot $V$ est un *facteur* d'un mot $U$ s'il existe deux autres mots $W$ et $W'$ de $\Sigma^*$ tels que $U = W \cdot V \cdot W'$. On dit aussi que $V$ est un *sous-mot* de $U$. On note $U_{[i,j[}$, où $i$ et $j$ sont deux entiers tels que $0 \leq i \leq j \leq |U|$, le facteur de $U$ commençant à l'indice $i$ de $U$ inclus, et finissant à l'indice $j$ de $U$ exclus. Par convention, si $i = j$, ce facteur est le mot vide. L'ensemble des facteurs d'un mot $U$ est noté $Fact(U)$.

— Pour tout entier $j$, tel que $0 \leq j \leq |U|$, le facteur $U_{[0,j[}$ est un *préfixe* de $U$. C'est l'unique préfixe de longueur $j$.

— Pour tout entier $j$, tel que $0 \leq j \leq |U|$, le facteur $U_{[j,|U|[}$ est un *suffixe* de $U$. C'est l'unique suffixe de longueur $|U| - j$.

**Exemple 2.** *Soit* $U = \overset{0\,1\,2\,3\,4\,5\,6\,7}{CTAGTTAG}$, *alors* $U_{[2,6[} = AGTT$. $CTAG$ *est un préfixe de* $U$, *car* $CTAGTTAG = CTAG \cdot TTAG$. *C'est le préfixe* $U_{[0,4[}$. $TTAG$ *est un suffixe de* $U$, *car* $CTAGTTAG = CTAG \cdot TTAG$. *C'est le suffixe* $U_{[4,8[}$.

## 2.2 La recherche de motifs approchée

### 2.2.1 La distance de Levenshtein

La notion de *recherche approchée* suppose que l'on puisse décrire le motif de manière approchée et que l'on soit capable de mesurer l'approximation faite entre deux mots. Pour cela, nous allons utiliser la *distance de Levenshtein*, également appelée *distance d'édition*. Cette distance repose sur trois *opérations d'édition* élémentaires qui affectent un caractère à la fois : la substitution, l'insertion et la délétion.

— *substitution* : remplacement d'un caractère par un autre ;
— *insertion* : ajout d'un caractère ;
— *délétion* : suppression d'un caractère.

L'application de chacune de ces trois opérations correspond à une *erreur* entre deux mots.

**Définition 1.** *Soient $U$ et $V$ deux mots de $\Sigma^*$. La distance de Levenshtein entre $U$ et $V$, notée $Lev(U,V)$, est le nombre minimal d'opérations d'édition nécessaires pour transformer $U$ en $V$. La suite d'opérations appliquées s'appelle un* script d'édition *entre $U$ et $V$.*

**Exemple 3.** *Pour les deux mots $U = CTAGTTAG$ et $V = CTACGATAG$, on a $Lev(U,V) = 2$. En effet, pour passer du mot $\overset{0\,1\,2\,3\,4\,5\,6\,7}{CTAGTTAG}$ au mot $\overset{0\,1\,2\,3\,4\,5\,6\,7\,8}{CTACGATAG}$, il est nécessaire d'insérer un C entre l'indice 2 et 3 et de substituer le T à l'indice 4 par un A, ce qui fait deux opérations d'édition. Cela garantit $Lev(U,V) \leq 2$. Par ailleurs, on peut vérifier que tout script d'édition comprenant une seule opération d'édition ne permet pas de transformer $U$ en $V$. Il s'ensuit que $Lev(U,V) = 2$.*

*Afin de faciliter la visualisation du script d'édition, il est possible de le représenter sous forme d'alignement.*

```
C T A - G T T A G
| | |   |   | | |
C T A C G A T A G
```

*Un trait vertical | signifie une identité entre les 2 lettres, l'absence de trait vertical |*

*entre 2 lettres signifie une substitution, le caractère – signifie un gap, c'est-à-dire soit une insertion, soit une délétion selon la séquence de référence.*

À partir de la distance de Levenshtein, on peut définir formellement le problème de la recherche de motifs approchée.

**Définition 2.** *Soient $P$ et $T$ deux mots de $\Sigma^*$, et soit $k$ un entier positif ou nul. Une* occurrence *de $P$ dans $T$ avec une distance de Levenshtein d'au plus $k$ est un facteur $F$ de $T$ tel que $Lev(P,F) \leq k$.*

La recherche de motifs approchée consiste alors à identifier toutes les occurrences du motif $P$ dans le texte $T$. Cette définition peut connaitre quelques variations, notamment en présence d'occurrences chevauchantes. Quand $Lev(P,F) < k$, par exemple, il existe des occurrences chevauchantes non optimales par ajout d'insertions ou de délétions aux extrémités. Ce type d'occurrences n'est pas forcément pertinent, et est souvent volontairement ignoré.

### 2.2.2 La recherche par programmation dynamique

Nous venons de voir que le problème de la recherche de motifs approchée repose sur la définition de la distance de Levenshtein. Nous commençons donc par expliquer comment calculer de manière efficace, par programmation dynamique, la distance de Levenshtein entre deux mots, puis comment cet algorithme peut être adapté à la recherche de motifs dans un texte.

#### Calcul de la distance de Levenshtein par programmation dynamique

Reprenons les deux mots $U$ = CTAGTTAG et $V$ = CTACGATAG de l'Exemple 3. Pour reconstruire le script d'édition entre $U$ et $V$, regardons quelle forme celui-ci peut avoir, et plus particulièrement quelle est la dernière opération d'édition appliquée. Il y a trois possiblités :

1. le G final de $U$ est aligné avec le G final de $V$, ce qui correspond à une *identité*, auquel cas le calcul se poursuit en comparant les deux préfixes CTAGTTA pour $U$ et CTACGATA pour $V$. Si les deux caractères avaient été différents, cela aurait donné lieu à une *substitution* ;

2. le script se termine par une *délétion* du G final de $U$, et le calcul se poursuit en comparant le préfixe CTAGTTA de $U$ avec $V$ ;

3. le script se termine par une *insertion* du G final de $V$, et le calcul se poursuit en comparant $U$ avec le préfixe CTACGATA de $V$.

Dans les trois cas, il faut ensuite continuer en calculant la distance de Levenshtein entre deux préfixes de $U$ et $V$. Le problème va donc demander de calculer $Lev(U_{[0,i[}, V_{[0,j[})$ pour tous $i$ et $j$ tels que $0 \leq i < |U|$ et $0 \leq j < |V|$. A chaque étape de l'algorithme, on suppose donc que les valeurs $Lev(U_{[0,i-1[}, V_{[0,j-1[})$, $Lev(U_{[0,i[}, V_{[0,j-1[})$ et $Lev(U_{[0,i-1[}, V_{[0,j[})$ sont connues, et on va en déduire $Lev(U_{[0,i[}, V_{[0,j[})$. Ces trois cas que nous avons distingués pour la dernière opération du script d'édition se généralisent à toutes les positions $i$ et $j$ :

— les caractères $U_{[i]}$ et $V_{[j]}$ sont alignés entre eux. Dans ce cas, nous partons des préfixes $U_{[0,i-1[}$ et $V_{[0,j-1[}$ et nous ajoutons $U_{[i-1]}$ aligné avec $V_{[j-1]}$. Si $U_{[i-1]} = V_{[j-1]}$, cela ne crée aucune erreur supplémentaire, et on a $Lev(U_{[0,i[}, V_{[0,j[}) = Lev(U_{[0,i-1[}, V_{[0,j-1[})$. Si au contraire $U_{[i-1]}$ et $V_{[j-1]}$ sont distincts, il est nécessaire d'appliquer une substitution, ce qui ajoute une erreur : $Lev(U_{[0,i[}, V_{[0,j[}) = Lev(U_{[0,i-1[}, V_{[0,j-1[}) + 1$ ;

— il y a une délétion d'un caractère dans $U$ à l'indice $i$, c'est-à-dire que nous avançons d'un caractère dans $U$ sans bouger dans $V$. Nous partons donc des préfixes $U_{[0,i-1[}$ et $V_{[0,j[}$ et nous supprimons la lettre $U_{[i-1]}$. Dans ce cas, $Lev(U_{[0,i[}, V_{[0,j[}) = Lev(U_{[0,i-1[}, V_{[0,j[}) + 1$.

— il y a une insertion d'un caractère de $V$ à la position $i$ de $U$, c'est-à-dire que nous avançons d'un caractère dans $V$ sans bouger dans $U$. Nous partons donc des préfixes $U_{[0,i[}$ et $V_{[0,j-1[}$ et insérons le caractère $V_{[j-1]}$. Dans ce cas, $Lev(U_{[0,i[}, V_{[0,j[}) = Lev(U_{[0,i[}, V_{[0,j-1[}) + 1$,

À partir de là, $Lev(U_{[0,i[}, V_{[0,j[})$ est la valeur minimale entre les trois possibilités.

Maintenant que nous somme capables de calculer toutes les distances de Levenshtein pour les couples $(i,j)$ à partir de la distance des couples $(i-1, j-1)$, $(i, j-1)$ et $(i-1, j)$, il reste à aborder la question des valeurs initiales, et plus exactement des couples de la forme $(0,0)$, $(0,j)$ et $(i,0)$. La valeur pour le couple $(0,0)$ est égale à $Lev(U_{[0,0[}, V_{[0,0[})$, la distance d'édition entre deux chaînes vides :

c'est donc 0. Pour le couple $(0,j)$, sachant que $U_{[0,0[} = \epsilon$, pour passer de $\epsilon$ à $V_{[0,j[}$, il faut faire l'insertion des $j$ premières lettres de $V$. Donc $Lev(U_{[0,0[}, V_{[0,j[}) = j$. Pour le couple $(i,0)$, sachant que $V_{[0,0[} = \epsilon$, pour passer de $U_{[0,i[}$ à $\epsilon$, il faut faire la délétion des $i$ premières lettres de $U$. Donc $Lev(U_{[0,i[}, V_{[0,0[}) = i$.

Pour résumer, voici l'équation du calcul de la distance de Levenshtein entre deux mots $U$ et $V$.

$$
\begin{aligned}
&Lev(U_{[0,0[}, V_{[0,0[}]) = 0 \\
&Lev(U_{[0,0[}, V_{[0,j[}) = j \\
&Lev(U_{[0,i[}, V_{[0,0[}) = i \\
&Lev(U_{[0,i[}, V_{[0,j[}) = \min \begin{cases}
\textit{substitution} & \begin{cases} Lev(U_{[0,i-1[}, V_{[0,j-1[}) & \text{si } U_{[i-1]} = V_{[j-1]} \\ Lev(U_{[0,i-1[}, V_{[0,j-1[}) + 1 & \text{si } U_{[i-1]} \neq V_{[j-1]} \end{cases} \\
\textit{délétion} & Lev(U_{[0,i-1[}, V_{[0,j[}) + 1 \\
\textit{insertion} & Lev(U_{[0,i[}, V_{[0,j-1[}) + 1
\end{cases}
\end{aligned}
\tag{2.1}
$$

{?}

Afin de ne pas avoir à recalculer les résultats intermédiaires, il est possible de les conserver dans une matrice à remplir au fur et à mesure. Par exemple, pour calculer la distance entre CTACGATAG et CTAGTTAG, la matrice est remplie comme suit : pour calculer $Lev(\text{CTAC}, \text{CTA})$ il suffit de calculer : $min(Lev(\text{CTAC}, \text{CT}) + 1, Lev(\text{CTA}, \text{CT}) + 1, Lev(\text{CTA}, \text{CTA}) + 1) = min(3, 2, 1) = 1$.

| | | C | T | A | C | ... |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | |
| C | 1 | 0 | 1 | 2 | 3 | |
| T | 2 | 1 | 0 | 1 | 2 | |
| A | 3 | 2 | 1 | 0 | **?** | |
| G | 4 | 3 | 2 | 1 | | |
| ... | | | | | | |

Nous pouvons ainsi remplir toute la table.

| | | C | T | A | C | G | A | T | A | G |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| C | 1 | **0** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| T | 2 | 1 | **0** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| A | 3 | 2 | 1 | **0** | **1** | 2 | 3 | 4 | 5 | 6 |
| G | 4 | 3 | 2 | 1 | 1 | **1** | 2 | 3 | 4 | 5 |
| T | 5 | 4 | 3 | 2 | 2 | 2 | **2** | 2 | 3 | 4 |
| T | 6 | 5 | 4 | 3 | 3 | 3 | 3 | **2** | 3 | 4 |
| A | 7 | 6 | 5 | 4 | 4 | 4 | 3 | 3 | **2** | 3 |
| G | 8 | 7 | 6 | 5 | 5 | 4 | 4 | 4 | 3 | **2** |

La distance entre les deux mots se trouve dans le coin inférieur droit. Ainsi $Lev(\text{CTACGATAG}, \text{CTAGTTAG}) = 2$. Ce calcul se fait en temps $O(|U| \times |V|)$.

Nous pouvons remonter le chemin dans la matrice qui a permis d'obtenir cette distance. Ce chemin permet de retrouver l'alignement entre les deux séquences, opération d'édition par opération d'édition. On retrouve bien l'alignement précédent donné dans l'Exemple 3.

Cet algorithme a été redécouvert à de multiples occasions dans différents domaines. En bioinformatique, c'est Needleman et Wunsch [118] qui est référencé. En informatique, cette approche est connue sous le nom d'algorithme de Wagner-Fischer [170]. Cependant, des références plus anciennes font déjà part de cette approche [166].

### Application à la recherche de motifs approchée

Dans le cadre de la recherche d'occurrences d'un motif $P$ dans un texte $T$, nous cherchons à aligner tous les facteurs de $T$ avec $P$. L'algorithme précédent appliqué naïvement donnerait ainsi une complexité en $O(|P|^2 \times |T|)$. Toutefois, lors de la comparison de $P$ avec deux facteurs de $T$ successifs, démarrant aux positions $j$ et $j+1$ par exemple, de nombreux calculs sont effectués en double et pourraient être mutualisés. Sellers [146] a proposé de modifier l'algorithme de programmation dynamique, simplement en initialisant $Lev(P_{[0,0[}, T_{[0,j[}]) = 0$. Cela revient à dire que les positions présentes dans le texte $T$ avant l'occurrence de $P$ ne sont pas pénalisées par un coût d'insertion. Il existe une occurrence de

$P$ dans $T$ dès lors que le couple de positions ($|P|$,$j$) a une valeur inférieure ou égale à $k$ et cette occurrence se termine à l'indice $j$ dans le texte. Pour retrouver le début de l'occurrence, il suffit de parcourir le chemin dans la matrice.

| | | C | T | A | C | G | A | T | A | G |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| T | 2 | 1 | 0 | 1 | 1 | 1 | 2 | 1 | 2 | 2 |
| A | 3 | 2 | 1 | 0 | 1 | 2 | 1 | 2 | 1 | 2 |
| G | 4 | 3 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 1 |
| T | 5 | 4 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 2 |
| T | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 2 | 3 | 3 |
| A | 7 | 6 | 5 | 4 | 4 | 4 | 3 | 3 | 2 | 3 |
| G | 8 | 7 | 6 | 5 | 5 | 4 | 4 | 4 | **3** | **2** |

On obtient une complexité en temps de $O(|P| \times |T|)$. Quand le nombre maximal d'erreurs entre $P$ et $T$ est $k$, plusieurs améliorations ont été proposées pour arriver à une complexité en $O(k|T|)$ tout d'abord en temps moyen [164], puis dans le pire des cas [88], et ce dans un espace en $O(|P|^2)$ [45].

De manière alternative, la recherche d'une occurrence dans un texte peut être réalisée à l'aide d'un automate. Celui-ci peut être construit en déterminisant un automate non déterministe capable de reconnaître les mots à une distance d'au plus $k$, ou en reprenant la matrice de programmation dynamique [164], chaque ensemble de valeurs composant une colonne dans cette matrice devient un état. Ces deux approches sont assez similaires. Le principal problème de cette approche va être le nombre d'états qui, tel que le proposait déjà Ukkonen était de $O(min(3^m, m(2m\sigma)^k))$, mais propose en contre-partie une complexité en temps de $O(n)$ [164] pour $k$ fixé.

Une dernière approche est la technique de bit-parallélisme. Cette approche tire profit de l'architecture des ordinateurs à travailler avec des mots machines, c'est-à-dire des vecteurs de bits d'une taille prédéfinie $w$, généralement 32 ou 64 bits aujourd'hui, pour paralléliser les calculs. Ainsi, les différentes techniques de bit-parallélisme vont être soit plus à rapprocher de la programmation dynamique soit plus des techniques d'automate.

### 2.2.3 Le filtrage par graine

Les approches par programmation dynamique pure présentées dans les paragraphes précédents ont pour inconvénient majeur d'être trop coûteuses en temps de calcul quand le texte $T$ est de grande taille. En effet, elles vérifient si chaque position de $T$ est le début d'une occurrence potentielle. Les *approches par filtrage* proposent d'éliminer sur la base de critères simples des régions de $T$ trop éloignées du motif $P$, et de ne vérifier ainsi que les positions de $T$ présentes dans des régions suffisamment similaires avec une région de $P$ pour contenir une occurrence. On recherche ainsi les occurrences de $P$ à partir de petits alignements locaux entre $P$ et $T$, qui servent en quelque sorte de points d'ancrage. Ces petits alignements locaux sont appelés des *graines*. Formellement, un critère de *filtre* est un prédicat $\mathcal{P}$ sur $\Sigma^* \times \Sigma^*$, tel que son évaluation sur deux mots $U$ et $V$ de $\Sigma$ est en lien avec le fait que la distance de Levenshtein entre $U$ et $V$ est inférieure ou égale à $k$.

Une approche par filtre procède typiquement en deux étapes.

1. *Filtrage* : identification des graines communes entre le texte $T$ et le motif $P$ ;
2. *Vérification* : pour chaque graine trouvée, vérification si la région concernée contient une occurrence. Cette étape peut se faire avec l'un des algorithmes de calcul de distance présentés en Section 2.2.2.

L'intérêt réside dans le fait que l'étape de filtrage peut être très rapide, en tirant partie du fait que la graine ne contient pas ou peu d'erreurs. En plus du temps calcul, les performances d'un filtre s'évaluent en termes de *pouvoir filtrant* et de *sensibilité*, qui à eux deux expriment le comportement attendu d'un bon filtre : sélectionner les positions qui correspondent à une occurrence, et écarter les autres. Supposons que le texte $T$ contienne $x$ occurrences du motif $P$ : $O_o, \ldots, O_{x-1}$.

— Le pouvoir filtrant est mesuré par le nombre de facteurs $F$ du texte $T$ tels que $\mathcal{P}(F,P)$ est vrai. Un mauvais pouvoir filtrant implique que l'algorithme va passer beaucoup de temps en vérification de candidats inutiles.

— La sensibilité est mesurée par la proportion des occurrences $O$ de $\{O_o, \ldots, O_{x-1}\}$ telles que $\mathcal{P}(O,P)$ soit vrai. Une mauvaise sensibilité implique que des

occurrences du motif seront perdues.

— Un filtre est *sans perte* si $\mathcal{P}(O,P)$ est vrai pour toute occurrence $O$ de $\{O_o,\ldots,O_{x-1}\}$. C'est donc un filtre de sensibilité maximale.

Cette méthode de « graine et extension » est maintenant massivement utilisée pour résoudre le problème de recherche approchée. La définition d'une graine peut être diverse et dépend de la nature du problème (taille de l'alphabet, taux d'erreur, ...). Nous donnons deux grands types d'exemples : les graines par facteur et les graine suffixes, qui se déclinent elles-mêmes en graines exactes ou approchées.

### Filtre par facteur

Le premier type de filtre utilise comme graine un facteur du motif, qui peut être exact ou approché.

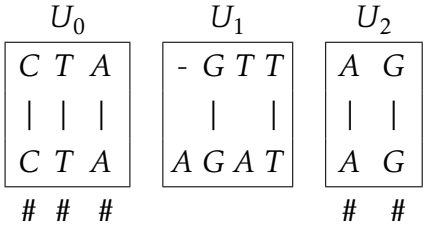
**Graine contigüe exacte** Le filtre par facteur le plus simple est défini à partir d'un ensemble de sous-mots exacts issus du motif. On parle de *graine contigüe exacte*. Supposons que nous ayons à chercher le motif $P$ avec au plus $k$ erreurs. D'après un raisonnement analogue au principe des tiroirs et des chaussettes, si on divise le motif $P$ en $k+1$ parties, c'est-à-dire en $k+1$ facteurs disjoints, il est certain que toute occurrence de $P$ contiendra au moins une de ces parties sans erreur.

**Lemme 1.** *Soit $U$ un mot de $\Sigma^*$ muni d'une factorisation en $k+1$ facteurs non vides : $U = U_0 \ldots U_k$. Pour tout mot $V$ de $\Sigma^*$ tel que $Lev(U,V) \leq k$, il existe $i$, $0 \leq i \leq k$, tel que $U_i$ est un facteur de $V$.*

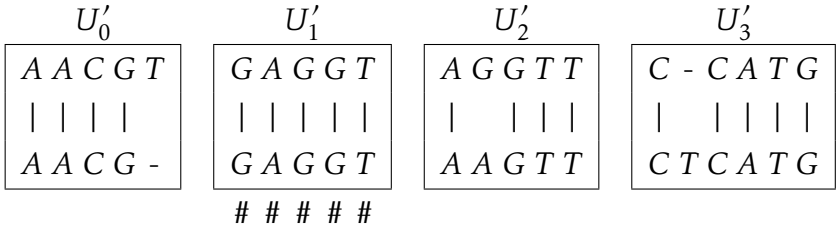
Cette propriété trouve une application pratique immédiate pour la recherche de motifs approchée : on découpe le motif $P$ en $k+1$ facteurs, qui vont servir de graines, et on cherche chacune de ces $k+1$ graines dans le texte.

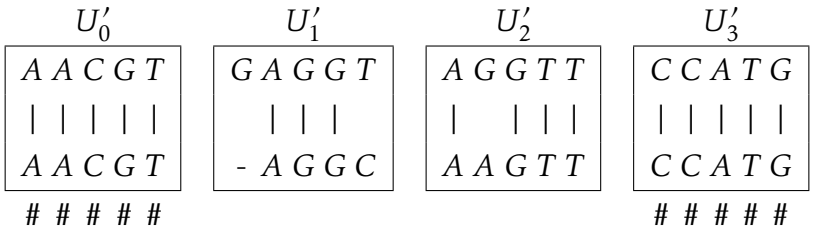
**Exemple 4.** *Reprenons les deux mots $U$ et $V$ de l'Exemple 3, utilisés ensuite pour illustrer le déroulement de l'algorithme par programmation dynamique pour le calcul de la distance de Levenshtein : $U$ = CTAGTTAG et $V$ = CTACGATAG. $U$ est de longueur 8. Si on cherche les occurrences de $U$ avec 2 erreurs, le Lemme 1 indique*

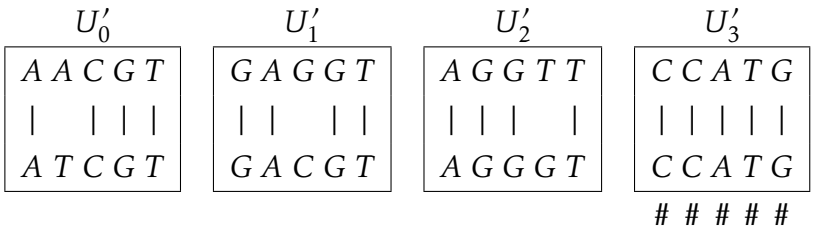
*que $U$ doit être découpé en trois facteurs, que nous choisissons ici de longueur 3, 3 et 2 : $U_0 = CTA$, $U_1 = GTT$ et $U_2 = AG$. Le filtre induit par cette partition de $U$ accepte bien $V$, avec deux facteurs en commun, $U_0$ et $U_2$.*




**Exemple 5.** *Considérons maintenant $U'$ =* AACGTGAGGTAGGTTCCATG*, de longueur 20, et un nombre d'erreurs $k = 3$. Cette fois, $U$ doit être découpé en quatre facteurs, que nous choisissons de longueur égale : $U'_0$ =* AACGT*, $U'_1$ =* GAGGT*, $U'_2$ =* AGGTT *et $U'_3$ =* CCATG*. Si on compare $U'$ aux trois mots $V'$ =* AACGGAGGTAAGTTCTCATG*, $W'$ =* AACGTAGGCAAGTTCCATG *et $Z'$ =* ATCGTGACGTAGGGTCCATG*, qui sont chacun à distance 3 de $U'$, il existe bien une occurrence d'une de ces quatre parties à chaque fois.*






Plus le nombre d'erreurs est important, plus les parties seront courtes et donc, plus le nombre d'occurrences potentielles à vérifier sera important. Ainsi l'efficacité du filtrage est d'autant moins importante que le taux d'erreur est élevé. Dans ce contexte, certains algorithmes de filtrage vont faire le choix de ne pas être exhaustif et d'améliorer le pouvoir filtrant au détriment de la sensibilité en prenant une graine plus longue. C'est ce qui est fait avec Blast, par exemple, avec une graine exacte contigüe chevauchante qui dans la plupart des applications entraine une perte d'occurrences.

**Les graines espacées** La limitation des graines contigües exactes en terme de sensibilité peut être partiellement contournée en utilisant des graines espacées qui offrent un meilleur compromis entre la sélectivité et la sensibilité. Une graine espacée est définie comme un sous-mot qui accepte une substitution à certaines positions fixes [74, 101].

**Définition 3.** *Une* graine espacée *est un mot sur l'alphabet* $\{\#, -\}$*, où le symbole* # *figure une identité, et* − *une identité ou une substitution.*

En pratique, une graine espacée est cherchée de manière chevauchante dans le motif, à la Blast. Pour chaque longueur de motif et chaque nombre maximal d'erreurs $k$, il est alors possible de concevoir une graine de poids optimal qui induit un filtre sans perte. Le logiciel Iedera fait cela par exemple [81].

**Exemple 6.** *On reprend le mot U' de l'Exemple* 5*, à comparer à Z', qu'il est possible d'aligner avec U' avec trois substitutions. Pour un motif de longueur* 20 *avec* 3 *substitutions, il existe une graine espacée de poids* 6 *qui permet un filtrage sans perte. Cette graine est* #-#--#-#--#-#*, déterminée avec Iedera. De fait, cette graine est présente dans l'alignement. Elle est même présente à quatre positions différentes.*

```
AACGTGAGGTAGGTTCCATG
| ||||| ||||| ||||||
ATCGTGACGTAGGGTCCATG
#-#---#-#---#-#
  #-#---#-#---#-#
   #-#---#-#---#-#
    #-#---#-#---#-#
```

*Comme cette graine est de poids 6, elle offre un meilleur pouvoir filtrant que la graine contigüe exacte #####, qui est de poids 5.*

Le principe des graines espacées a été étendu aux *graines sous-ensembles*, qui permettent de distinguer les cas où le caractère - couvre une transition ou une transversion [82]. Il est également intéressant de combiner plusieurs graines, graines espacées ou graines sous-ensembles, dans le cadre de la recherche d'homologie. Mais, dans notre contexte de recherche de motif, les graines espacées n'améliorent pas la sensibilité par rapport à des graines contigües lorsqu'elles sont utilisées avec des insertions ou des délétions. Une tentative de généraliser les graines espacées à une distance de Levenshtein a été réalisée avec une forte restriction sur le nombre total d'erreurs.

**Les graines avec erreurs** Une autre approche est l'approche *hybride* qui combine le filtrage par graine et la recherche avec erreurs. Comme pour la graine exacte contigüe, le motif est divisé en parties non chevauchantes, mais chaque partie peut être recherchée avec un nombre donné d'erreurs.

**Lemme 2.** *Soit $U$ un mot de $\Sigma^*$, partionné en $s$ facteurs $U_0, \ldots, U_{s-1} : U = U_0 \ldots U_{s-1}$. Soit également une suite de $s$ entiers strictement positifs $t_0, \ldots, t_{s-1}$ dont la somme est égale à $k+1$. Pour tout mot $V$ de $\Sigma^*$, si $Lev(U,V) \leq k$, alors il existe $i$, $0 \leq i \leq s-1$, et un facteur $F$ de $V$ tels que $Lev(U_i, F) < t_i$.*

On retrouve le Lemme 1 en considérant le cas particulier $s = k+1$ et $t_0 = \ldots = t_{s-1} = 1$. De manière générale, les approches hybrides différent dans la division du motif et le nombre maximal d'erreurs autorisé dans chacune d'elle. Par exemple, Navarro et Baeza-Yates [116] ont conçu une méthode hybride qui

consiste à diviser le motif $P$ en $s = \frac{|P|+k}{\log_\sigma |T|}$ parties, où $\sigma$ est la taille de l'alphabet, et de rechercher ces parties avec $\lfloor \frac{k}{s} \rfloor$ erreurs. Cette approche a été aussi utilisée avec différents index (LZ-index et le FM-index) [137]. L'approche hybride est plus flexible que les graines contigües ou espacées, car elle tolère n'importe quel type d'erreur sans restriction de nombre ou de position. Cependant, leur pouvoir filtrant tend à décroître lorsque que le nombre d'erreurs croît. Encore une fois, cette approche est pratique pour un taux d'erreur faible.

**Filtre par suffixe**

À côté du filtrage par facteur, il est possible de chercher à allonger la taille de la graine sur laquelle va porter le filtre, en cherchant non plus un facteur mais un *suffixe* du motif. C'est ce qui est proposé dans [73] par Kärkkäinen et Na. Comme pour le Lemme 2, on suppose que le mot $U$ est muni d'une factorisation en $s$ parties, et que l'on a une suite de $s$ entiers $t_0, \ldots, t_{s-1}$.

**Lemme 3.** *Soit $U$ un mot de $\Sigma^*$, partionné en $s$ facteurs $U_0, \ldots, U_{s-1} : U = U_0 \ldots U_{s-1}$. Soit également une suite de $s$ entiers strictement positifs $t_0, \ldots, t_{s-1}$ dont la somme est égale à $k+1$. Pour tout mot $V$ de $\Sigma^*$, si $Lev(U,V) \leq k$, alors il existe $i$, $0 \leq i \leq s-1$, tel que pour tout $j$, $i \leq j \leq s-1$, on a $Lev(U_i \ldots U_j) \leq t_i + \cdots + t_j$.*

Ce lemme indique qu'il est possible de trouver non pas un facteur approché, mais une suite de facteurs approchés formant un suffixe approché commun entre $U$ et $V$. En outre, les erreurs au sein de ce suffixe commun ne sont pas distribuées aléatoirement, mais régulièrement. Dans [73], il est montré expérimentalement que les meilleures performances en terme de pouvoir filtrant étaient réalisées avec $s = k+1$ et $t_0 = \ldots = t_k = 1$.

**Exemple 7.** *Nous reprenons les séquences $U'$, $V'$, $W'$ et $Z'$ de l'Exemple 5, avec la factorisation en 4 facteurs $U'_0, \ldots, U'_3$ pour $U'$. Pour cette factorisation, on choisit la suite $t$ définie par $(t_0, \ldots, t_3) = (1, \ldots, 1)$. Chaque tableau ci-dessous indique la distance de Levenshtein pour chaque couple de blocs. Le bloc correspondant à l'indice $i$ dans le Lemme 3 est encadré. Ainsi, pour $V'$ le suffixe commun est $U'_I U'_2 U'_3$, alors que pour $W'$ et $Z'$ il s'agit de $U'_3$.*

| | $U'_0$ | $U'_1$ | $U'_2$ | $U'_3$ |
|---|---|---|---|---|
| $t$ | 1 | 1 | 1 | 1 |
| $V'$ | 1 | 0 | 1 | 1 |

| | $U'_0$ | $U'_1$ | $U'_2$ | $U'_3$ |
|---|---|---|---|---|
| $t$ | 1 | 1 | 1 | 1 |
| $W'$ | 0 | 2 | 1 | 0 |

| | $U'_0$ | $U'_1$ | $U'_2$ | $U'_3$ |
|---|---|---|---|---|
| $t$ | 1 | 1 | 1 | 1 |
| $Z'$ | 1 | 1 | 1 | 0 |

## 2.3 L'indexation plein texte

L'utilisation de graines pour la recherche de motifs approchée nécessite de pouvoir trouver rapidement les occurrences d'une graine dans un texte. Une approche simple consiste à extraire toutes les graines possibles du texte et à enregistrer les positions de leurs occurrences dans une table. Cependant un tel index dépendra de la graine et ne sera d'aucune aide pour une autre graine.

Nous appelons *indexation plein texte* le fait d'indexer, et donc de pouvoir rechercher, n'importe quelle sous-chaîne d'un texte. Cette solution offre une plus grande souplesse d'utilisation et est également potentiellement plus économe en espace qu'une table de hachage. Ces avantages se font au détriment d'un temps d'accès plus lent que pour ces dernières.

### 2.3.1 Pourquoi indexer ?

Pour rechercher une information précise au sein de données, une solution simple consiste à les parcourir en totalité. Lorsque un grand nombre de requêtes est réalisé, chacune d'entre elle va nécessiter de re-parcourir la totalité des données. Cela devient rapidement rédhibitoire dès que le nombre de requêtes est important.

Un index est une structure de données permettant de répondre à de telles requêtes, sans avoir à parcourir les données dans leur intégralité.

Pour un texte, un index permet généralement deux types de requêtes : *compter* le nombre d'occurrences d'une suite quelconque de caractères, et *localiser* chacune de ces occurrences. Un index permet de répondre rapidement à ces requêtes sans avoir à parcourir l'intégralité du texte. Cependant, construire un index a un coût en temps. Sachant que l'index porte sur le texte entier, sa construction nécessite au minimum de parcourir le texte intégralement. Construire un index pour une seule recherche n'est donc pas intéressant. L'indexation va être utile dès que le texte est connu à l'avance et qu'un nombre suffisamment important de requêtes vont devoir être faites. L'index permet d'avoir des requêtes dont le temps est généralement indépendant, ou dépendant logarithimiquement, de la longueur du texte alors qu'en absence d'index ces temps sont linéaires dans le

pire des cas. Cependant, pour garantir son efficacité, un index est généralement stocké en mémoire centrale et il a un coût non négligeable en espace.

Dans cette section nous nous intéresserons uniquement aux index dit plein texte. À l'inverse des listes inversées ou des tables de hachage, par exemple, ces index permettent de chercher un facteur quelconque d'un texte. Nous commençons par présenter la terminologie de cette famille d'index.

**Définition 4.** *Un index succinct est un index dont la taille est proportionnelle à l'espace théoriquement minimal pour stocker tout texte de même taille et de même alphabet que celui indexé.*

**Définition 5.** *Un index compressé, tire parti de la régularité du texte indexé, il prend un espace mémoire proportionnel à l'entropie empirique du texte.*

**Définition 6.** *Un auto-index est un index compressé qui contient suffisamment d'information pour pouvoir reconstituer n'importe quel facteur du texte indexé. Un auto-index peut donc se substituer au texte.*

Nous allons maintenant présenter les index plein texte les plus courants.

### 2.3.2 Un premier index, le trie des suffixes.

**Définition 7.** *Un trie des suffixes d'un texte $T$ est un trie contenant l'ensemble des suffixes de $T$.*

Le trie des suffixes est aussi nommé arbre des suffixes. En partant de la racine, il permet d'accéder à tous les suffixes et lire uniquement leur préfixe revient à parcourir les facteurs de $T$. Supposons que passer d'un nœud $u$ du trie des suffixes à un nœud $v$, enfant de $u$, se fasse en temps constant. À partir de la racine, parcourir le chemin correspondant à un motif $P$, facteur de $T$ avec $|P| = m$, se fait en temps $O(m)$. Si $P$ n'est pas un facteur de $T$ alors il n'existe pas de chemin correspondant à $P$ partant de la racine. Ainsi, il est très simple et très rapide (en temps $O(m)$) de vérifier si le motif $P$ est un facteur de $T$. Dans un trie des suffixes, la position du début du suffixe est conservée dans les feuilles. Pour connaître les positions d'un facteur, il suffit donc de regarder les feuilles présentes dans le sous-arbre du nœud associé à ce facteur Figure 2.1.

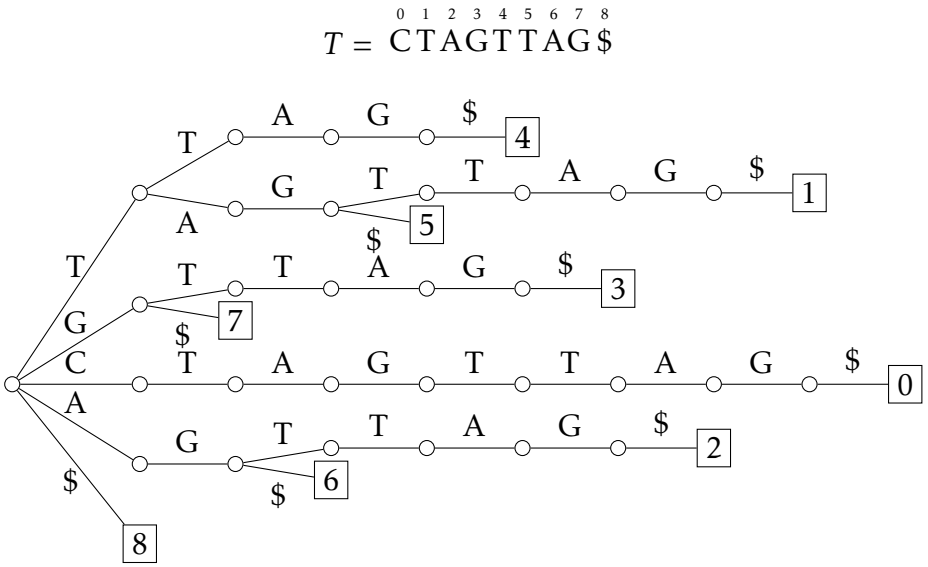



FIGURE 2.1 – Le trie des suffixes (ou l'arbre des suffixes) pour le mot CTAGTTAG$. Tout chemin partant de la racine (à gauche) vers une feuille (représentée par un rectangle) est un suffixe du mot CTAGTTAG$. La valeur dans une feuille est l'indice de début du suffixe dans le texte.

**Exemple 8.** *Dans la Figure 2.1, il existe un chemin étiqueté par TAG partant de la racine allant jusqu'à un nœud. Les feuilles présentes dans ce sous-arbre sont les feuilles 1 et 5, ce qui signifie que le motif TAG apparaît en positions 1 et 5 dans le texte.*

### 2.3.3 L'arbre compact des suffixes

L'arbre des suffixes peut être composé de longs chemins unaires : tous les nœuds intermédiaires sur ce chemin ont exactement un seul enfant. Cela conduit à un gaspillage de l'espace mémoire à cause du trop grand nombre de branches et de nœuds dans l'arbre. Dans le pire des cas le nombre de nœuds et de feuilles peut être quadratique (dans le cas où le texte est composé de caractères tous distincts par exemple). Une telle complexité en espace est incompatible avec l'indexation de grands textes. L'*arbre compact des suffixes* permet de solutionner ce problème. Il a été introduit par Weiner [174] et des recherches ultérieures ont permis d'améliorer son algorithme de construction [110, 163].

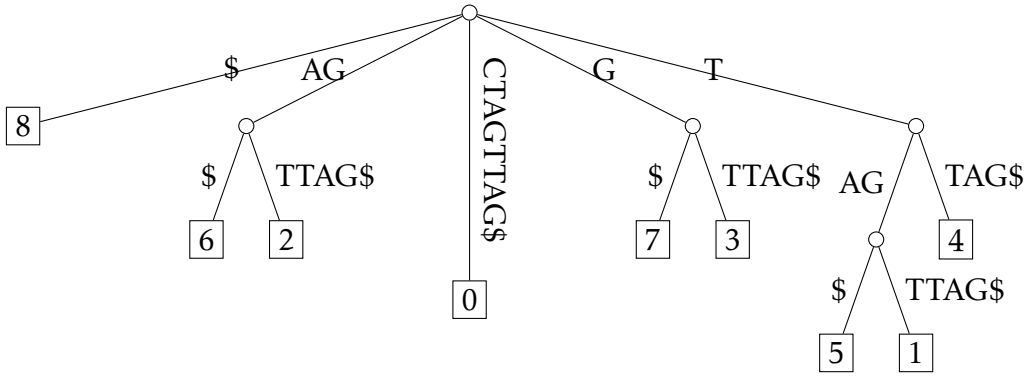



FIGURE 2.2 – L'arbre compact des suffixes pour le mot CTAGTTAG$. Chaque chemin unaire de l'arbre des suffixes est fusionné en une seule branche.

**Définition 8.** *L'arbre compact des suffixes d'un texte $T$ est un trie des suffixes dans lequel chaque chemin unaire est converti en une simple branche étiquetée par la concaténation des caractères remplacés.*

L'arbre compact des suffixes est une simplification du trie des suffixes dans lequel les nœuds ne possédant qu'un seul descendant sont retirés Figure 2.2.

L'arbre compact des suffixes d'un mot $T$ de longueur $n$ possède entre $n+1$ et $2n-1$ nœuds. En effet, l'arbre compact d'un texte $T$ possède au moins $n+1$ nœuds et feuilles correspondant aux $n+1$ suffixes de ce texte (le suffixe $\epsilon$ étant la racine). Dans le cas particulier d'un texte où chaque caractère est unique, il n'existe qu'un seul nœud (la racine) et $n$ feuilles correspondant aux $n$ suffixes non vides. Cela nous amène à la borne minimale de $n+1$ nœuds. Dans le pire des cas, chaque nœud a deux descendants, le nombre de feuilles étant fixés ($n$), il ne peut y avoir plus de $n$ nœuds ($n/2 + n/4 + \cdots + 1 < n$). Dans l'arbre compact des suffixes il y a donc $2n-1$ nœuds au maximum.

Cependant, même si le nombre de nœuds et feuilles est linéaire dans la taille du texte, les structures permettant de représenter un tel arbre ne sont pas linéaires en espace, elle sont en $O(n\log(n))$. On peut cependant trouver des articles indiquant une complexité en espace en $O(n)$ (par exemple [1]), cela se fait sous l'hypothèse qu'un mot machine peut être stocké en espace $O(1)$. Une telle structure peut être construite en un temps $O(n)$, dans un modèle de machine où les mots de taille $\log n$ peuvent être traités en temps constant. L'arbre compact

$T =$ C T A G T T A G $

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| SA | 8 | 6 | 2 | 0 | 7 | 3 | 5 | 1 | 4 |
| | $ | A | A | C | G | G | T | T | T |
| | | G | G | T | $ | T | A | A | T |
| | | $ | T | A | | A | G | G | A |
| | | | A | G | | G | $ | T | G |
| | | | G | T | | $ | | A | $ |
| | | | $ | A | | | | G | |
| | | | | G | | | | $ | |
| | | | | $ | | | | | |

FIGURE 2.3 – Table des suffixes pour le mot CTAGTTAG$. Les suffixes correspondants sont affichés verticalement et sont triés par ordre alphabétique. Seules les valeurs numériques sont réellement stockées, les suffixes ne sont affichés qu'à titre indicatif. SA[5] = 7 ce qui signifie que le 5e suffixe dans l'ordre alphabétique est celui en position 7 dans le texte, c'est-à-dire G$. La table des suffixes peut être aussi retrouvée à l'aide de l'arbre (compact ou non) des suffixes.

des suffixes permet de compter les occurrences d'un motif en temps optimal $O(m)$ en suivant un principe semblable à celui du trie des suffixes : il suffit de parcourir le chemin partant de la racine et étiqueté par le motif. Retrouver les positions des occurrences du motif se fait également en temps optimal $O(m+occ)$ où $occ$ est le nombre d'occurrences du motif.

Les complexités en temps des requêtes sur l'arbre des suffixes sont optimales, signifiant qu'il n'est pas possible de trouver d'algorithmes asymptotiquement plus rapides en théorie. L'inconvénient de l'arbre des suffixes réside dans l'espace mémoire qui lui est nécessaire : $10n$ octets en moyenne [84].

### 2.3.4 La table des suffixes

Pour remédier à l'inconvénient de l'arbre compact des suffixes, une structure plus économe a été introduite, la *table des suffixes* [12, 108].

**Définition 9.** *La table des suffixes d'un texte $T_{1,n}$ est une table $SA[1,n]$ contenant une permutation de l'intervalle $[1,n]$ telle que $T_{[SA[i],n[} < T_{[SA[i+1],n[}$ pour tout $1 \leq i < n$.*

Autrement dit, la table des suffixes d'un texte $T$ est une table contenant la position dans $T$ des suffixes triés dans l'ordre alphabétique (voir FIGURE 2.3). Elle

peut être construite en listant de gauche à droite les feuilles de l'arbre (compact) des suffixes. D'autres algorithmes plus sophistiqués permettent sa construction en un temps linéaire sans recourir à l'arbre des suffixes [71, 76, 77]. Cependant les algorithmes les plus efficaces en pratique ont une complexité dans le pire des cas en $O(n^2 \log n)$ [127].

Stocker un entier quelconque dans l'intervalle [1,$n$] peut se faire en $\log(n)$ bits. Une table de suffixes peut donc être représentée dans un espace de $n\log(n)$ bits. En pratique, si $n < 2^{32}$, la table des suffixes est stockée sur $4n$ octets.

La table des suffixes et le texte sont suffisants pour retrouver les occurrences d'un motif. Les suffixes partageant le même préfixe sont consécutifs dans la table, car triés dans l'ordre alphabétique. Il suffit alors d'utiliser une recherche dichotomique dans la table pour retrouver les bornes de cet intervalle. Dans une recherche dichotomique, il y a $O(\log n)$ étapes, chacune prenant au pire un temps linéaire dans la taille du motif. Une telle recherche s'effectue donc en un temps $O(m\log(n))$ dans le pire des cas, pour le comptage, et en $O(m\log(n) + occ)$ pour identifier les positions des occurrences.

#### Amélioration du temps de recherche

Il est aussi possible d'ajouter une table qui permettra d'améliorer la complexité en temps des requêtes afin de s'approcher de l'optimalité, la *table des plus longs préfixes communs*.

**Définition 10.** *La table des plus longs préfixes communs, de longueur $n-1$, stocke pour tout indice $0 \leq i < n-1$, la longueur du préfixe commun entre le suffixe commençant en position $SA[i]$ et celui commençant en position $SA[i+1]$ dans $T$.*

À l'aide de cette table, il est possible de réduire le temps de recherche à $O(m + \log(n))$. Une version compacte permet même de conserver une taille de structure de l'ordre de $O(n\log(n))$.

Les arbres compacts des suffixes comme les tables de suffixes ne sont pas des index succincts. Pour cela il faudrait atteindre une complexité en $n\log\sigma + o(n\log\sigma)$, où $\sigma$ est la taille de l'alphabet. Des travaux ont été menés afin de diminuer l'espace utilisé par de telles structures.

**Vers des tables compressées**

Il existe des régularités dans la table des suffixes dont il est possible de tirer parti afin de diminuer l'espace consommé.

**Exemple 9.** *En reprenant l'exemple de la Figure 2.3, nous pouvons voir une forme de régularité : les valeurs 7 et 3 sont consécutives dans la table des suffixes, tout comme 6 et 2 ainsi que 5 et 1. Il y a à chaque fois une différence de 4 entre la première valeur et la seconde. Ces régularités proviennent de la répétition du facteur TAG dans le texte.*

Sur de grands textes, tels des génomes, les répétitions sont plus longues et produisent alors des régularités plus importantes qui conduisent à de substantiels gains d'espace. C'est l'idée qui réside derrière le concept de table compacte des suffixes et de table compacte et compressée des suffixes [104, 105].

Une autre piste de compression repose sur une fonction $\Psi$, définie à partir de la table des suffixes.

**Définition 11.** *Soit une table des suffixes* $SA[1,n]$.
$\Psi : [1,n] \longrightarrow [1,n]$ *est telle que* $\forall i \in [1,n-1]$, $SA[\Psi(i)] = SA[i]+1$, *et* $SA[\Psi(n)] = 1$.

De manière générale, $\Psi(i)$ donne la position dans la table des suffixes du suffixe qui suit le suffixe en position $\mathsf{SA}[i]$ dans le texte. Cette fonction a la propriété d'être composée de, au plus, $\sigma$ blocs croissants, où $\sigma$ désigne la taille de l'alphabet. Une telle propriété permet de compresser efficacement la fonction $\Psi$. Cette propriété est utilisée dans les tables compressées des suffixes de Grossi et Vitter [53] et de Sadakane [139].

Parmi ces différentes structures d'indexation, seules celles de Mäkinen et Navarro [105] et de Sadakane [139] sont des auto-index, évitant de stocker le texte en plus de la structure.

### 2.3.5 Le FM-Index

Une autre piste de recherche dans le domaine des structures d'indexation compressées a consisté à explorer un concept très proche de la table des suffixes, mais qui était uniquement utilisé en compression de données : la transformée de Burrows-Wheeler. Cette transformée est associée par Ferragina et Manzini [42] à des structures additionnelles qui permettent d'aboutir à un auto-index.

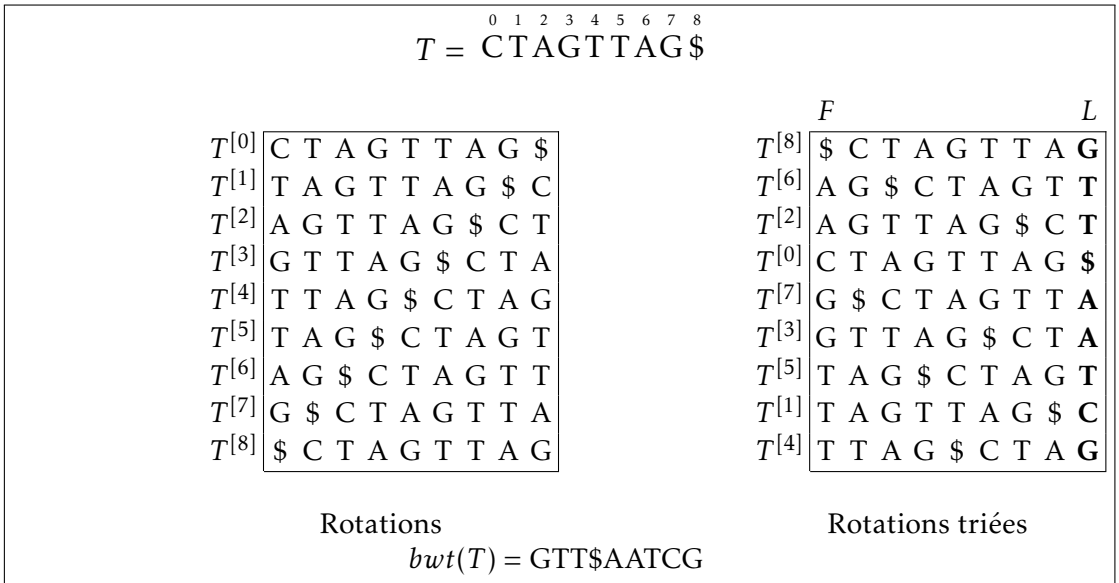



FIGURE 2.4 – Construction de la transformée de Burrows-Wheeler pour le mot CTAGTTAG$. Les rotations du mot sont d'abord triées, la transformée de Burrows-Wheeler correspond à la dernière colonne (marquée $L$)

**La transformée de Burrows-Wheeler**

La transformée de Burrows-Wheeler d'un texte [21] consiste en une réorganisation de ce texte afin de le compresser plus efficacement. De ce fait, elle est utilisée dans l'utilitaire de compression `bzip2`. Pour présenter cette transformée, nous commençons par introduire le concept de rotation.

**Définition 12.** *Une rotation d'un texte $T$ de longueur $n$ est une chaîne de caractères de même longueur composée d'un suffixe de $T$ concaténé à un préfixe de $T$. Nous notons $T^{[k]}$ la $k$-ème rotation du texte $T$, c'est-à-dire celle dont le préfixe de $T$ est de longueur $k$.*

**Exemple 10.** *Soit $T = \overset{0\,1\,2\,3\,4\,5\,6\,7}{CTAGTTAG}$, alors $T^{[3]} = GTTAGCTA$.*

Pour obtenir la transformée de Burrows-Wheeler d'un texte $T$, il suffit de trier dans l'ordre alphabétique les $n$ rotations de $T$ puis de concaténer la dernière lettre de chacune de ces rotations triées (voir FIGURE 2.4). Cette transformée se compresse mieux que le texte initial car les lettres identiques vont être plus souvent consécutives. En effet il y a de fortes chances pour que des rotations

consécutives (dans l'ordre alphabétique) soient précédées par une même lettre. Ce sont ces lettres qui se retrouvent dans la transformée de Burrows-Wheeler.

En raison de la proximité entre le concept de la transformée de Burrows-Wheeler (qui repose sur le tri de rotations) et de la table des suffixes (qui repose sur le tri des suffixes) [138], il est possible de définir la transformée de Burrows-Wheeler d'un texte $T$, BWT, à partir de la table des suffixes SA de ce texte.

**Définition 13.** *$BWT[i] = T[SA[i]-1]$, pour $SA[i] > 0$; $BWT[i] = T[n-1]$, sinon.*

Cette méthode de construction est très simple mais présente un écueil important : elle nécessite la construction de la table des suffixes dont on considère qu'elle prend trop d'espace. D'autres solutions, reposant sur la construction par bloc de la table des suffixes ou utilisant des propriétés de la transformée de Burrows-Wheeler ont permis de restreindre l'espace mémoire utilisé pendant la construction [26, 72, 96, 147].

L'inversion de la transformée de Burrows-Wheeler est rendue possible grâce à la fonction LF.

**Définition 14.** *Soit $T^{[k]}$, $k > 0$, la $i+1$-ème rotation dans l'ordre alphabétique. $LF(i) = j$ si et seulement si $T^{[k-1]}$ est la $j+1$-ème rotation dans l'ordre alphabétique.*

Si l'on considère les rotations triées comme étant dans un tableau, chaque ligne correspondant à une rotation, alors la dernière colonne $L$ de cette matrice est la transformée de Burrows-Wheeler. Nous appelons $F$ la première colonne. La fonction LF permet ainsi de passer d'une rotation à sa précédente dans ce tableau imaginaire. Cela revient à passer d'une lettre dans la colonne $L$ à cette même lettre (au même indice dans le texte) dans la colonne $F$.

**Exemple 11.** *Dans la Figure 2.4, prenons $i = 1$ correspondant à l'indice de la rotation $T^{[6]}$ dans le tableau, L'indice de la rotation $T^{[5]}$ est 6, donc $LF(1) = 6$. En regardant la lettre T dans la colonne L de la rotation $T^{[6]}$ correspond bien à la mettre lettre T dans la colonne F de la rotation $T^{[5]}$.*

Cette fonction LF est en fait l'inverse de la fonction $\Psi$, définie pour les tables compressées des suffixes. Cependant à l'inverse de la fonction $\Psi$, la fonction LF n'a pas besoin d'être stockée. Une valeur quelconque de la fonction peut être

calculée directement à partir de la transformée de Burrows-Wheeler et cette fonction peut être utilisée pour rechercher des motifs. C'est ce qu'ont conçu Ferragina et Manzini pour leur FM-index [42].

**Structures additionnelles**

Le calcul de la fonction LF, à partir de la colonne $L$, se fait de la façon suivante :

$$\mathsf{LF}(i) = \mathsf{rank}_{L[i]}(L, i) + C[L[i]] - 1$$

Où $\mathsf{rank}_{L[i]}(L, i)$ renvoie le nombre d'occurrences de $L[i]$ dans $L$ jusqu'à la position $i$ et $C[L[i]]$ correspond à la position de la première occurrence de $L[i]$ dans $F$. Après un pré-calcul sur $L$ la complexité en temps de la fonction rank est constante sur de petits alphabets [43, 52]. Une fois cette fonction LF définie, il est possible de rechercher des motifs avec la transformée de Burrows-Wheeler.

Supposons que nous connaissions l'intervalle $[i,j]$ d'un mot $u$, et que nous souhaitions retrouver l'intervalle du facteur $v = c \cdot u$, $c$ étant un symbole de l'alphabet. Pour cela il suffit de retrouver dans l'intervalle $[i,j]$ de la transformée de Burrows-Wheeler les positions di et fi de la première et de la dernière occurrence de $c$ et d'appliquer la fonction LF à ces deux positions. L'intervalle résultant $[\mathsf{LF}(\mathsf{di}), \mathsf{LF}(\mathsf{fi})]$ correspond à toutes les occurrences du mot $v$. Si $v$ n'apparaît pas dans le texte, il n'y a aucun $c$ dans l'intervalle $[i,j]$. La recherche d'un motif dans un FM-Index s'effectue de droite à gauche, par ajout successif de lettre en tête de motif (voir Figure 2.5).

Le FM-Index est bien un auto-index car il permet d'extraire n'importe quelle chaîne arbitraire du texte. En effet la transformée de Burrows-Wheeler étant inversible il est possible de retrouver le texte d'origine en utilisant la fonction LF. Le FM-index permet de compter le nombre d'occurrences d'un motif de taille $m$, et donc de vérifier son existence, en temps $O(m)$ et occupe un espace mémoire, dépendant de la compressibilité du texte, qui peut être inférieur à la taille du texte originel.

Puisqu'aucune information de position n'est directement stockée avec la transformée de Burrows-Wheeler, il n'est pas possible de savoir à quelles positions apparaissent les occurrences du motif dans le texte. Pour cela il est nécessaire

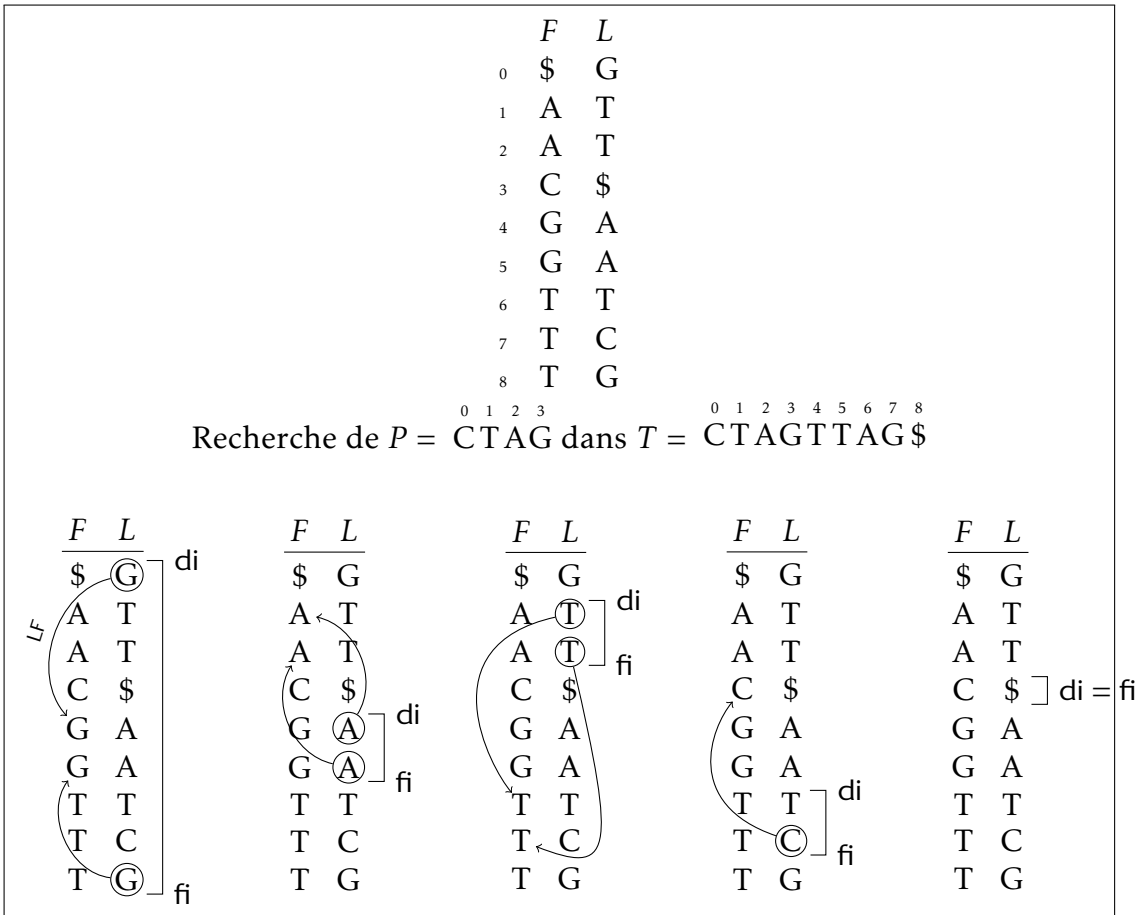



FIGURE 2.5 – Utilisation du FM-index du mot CTAGTTAG$ pour rechercher le motif CTAG. La colonne $L$ désigne la transformée de Burrows-Wheeler alors que la colonne $F$ correspond à la première lettre de toutes les rotations triées (il s'agit donc aussi de $L$ triée). Les suffixes G, AG, TAG puis CTAG sont successivement recherchés à partir du motif précédent. di et fi désignent respectivement le début et la fin de l'intervalle des occurrences du suffixe considéré à l'étape courante. Les flèches correspondent au résultat de la fonction LF pour la position entourée.

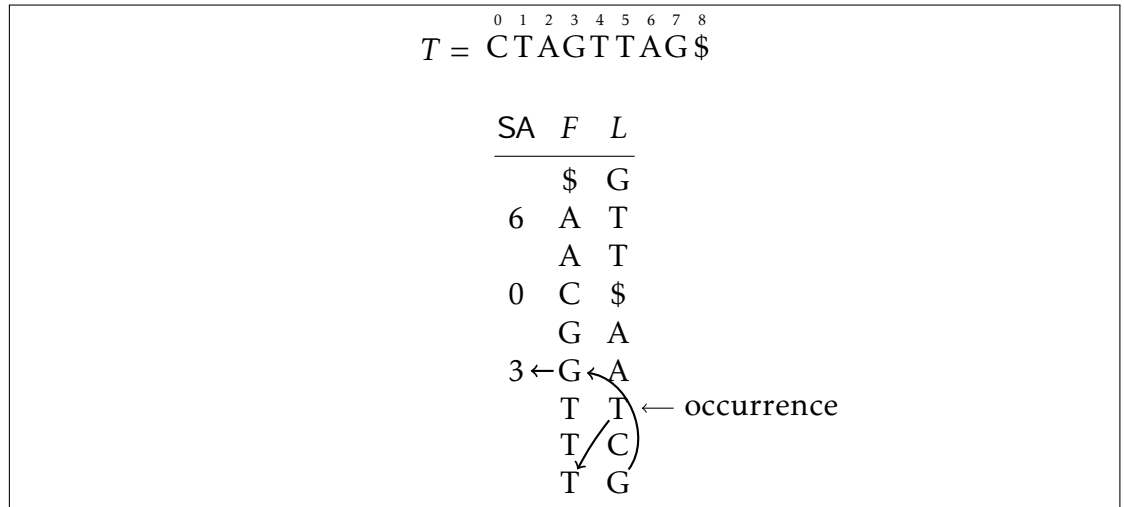



FIGURE 2.6 – Échantillonnage de la table des suffixes pour le FM-index. Supposons que l'on veuille connaître la position de l'occurrence. On part de celle-ci et on utilise la fonction LF jusqu'à arriver à une valeur échantillonnée.
En appliquant deux fois LF, on arrive sur la valeur échantillonnée 3, nous étions donc partis de la valeur 5. L'occurrence est dans en position 5 dans le texte.

d'ajouter un échantillon de la table des suffixes qui, elle, fournit les positions des occurrences. La table est échantillonnée afin de prendre un espace qui reste négligeable comparé à la transformée de Burrows-Wheeler. Lorsqu'on souhaite connaître la position dans le texte d'une rotation (c'est-à-dire la position à laquelle elle débute), deux situations se présentent. Soit la position est directement connue car elle est échantillonnée, soit ce n'est pas le cas. Dans le second cas, il faut utiliser la fonction LF autant de fois que nécessaire afin de tomber sur une rotation dont la position est échantillonnée. Une fois cela fait, il suffit d'ajouter à la position obtenue le nombre d'appels à la fonction LF pour connaître la position de la rotation d'intérêt (voir Figure 2.6). En théorie une position sur $\log^{1+\varepsilon} n$, $\varepsilon > 0$, est échantillonnée. Trouver toutes les occurrences d'un motif de longueur $m$ avec un FM-index prend alors un temps $O(m + occ \log^{1+\varepsilon} n)$.

**Structures d'indexation compressées en pratique**

Nous avons vu diverses structures d'indexation compressées, y compris des auto-index, dont l'intérêt en terme d'espace mémoire économisé est important.

Sur un génome humain de 3 milliards de nucléotides, il faut compter 15 Go pour stocker une table des suffixes contre 1 Go à 2 Go pour une structure d'indexation compressée telle qu'un FM-index.

Mais il y a un revers à la médaille. Bien qu'en théorie ces structures d'indexation compressées n'ont pas nécessairement des complexités en temps moins bonnes que la table des suffixes, elles sont en pratique environ deux ordres de grandeur plus lentes qu'une table des suffixes [44].

L'intérêt d'utiliser de telles structures reste réel : malgré ces temps de recherche beaucoup plus lents que la table des suffixes, la recherche d'un motif est très rapide comparée au parcours de la totalité du texte. De plus si la table des suffixes ne rentre pas complètement dans la mémoire centrale de la machine, les performances seront lourdement pénalisées rendant alors les structures d'indexation compressées plus rapides.

## 2.4 Les graines 01*0

### 2.4.1 Définitions

Nous avons déjà souligné qu'en utilisant un principe similaire au principe des tiroirs, il est évident que si $P$ est partitionné en $k+1$ parties, alors chaque mot $U$, tel que $ed(P,U) \leq k$, contient au moins une de ces parties. De manière similaire, si $P$ est partitionné en $k+2$ parties, notées $P_1,\ldots,P_{k+2}$, alors $U$ devrait contenir deux parties de $P$. Les parties ne font pas nécessairement la même longueur. Le lemme suivant permet de définir des critères plus stricts : il existe deux parties conservées qui sont séparées par des parties contenant exactement une erreur.

**Lemme 4.** *Soit $U \in \Sigma^*$ tel que $ed(P,U) \leq k$. Il existe $i,j$, $1 \leq i < j \leq k+2$, et $U_1,\ldots,U_{j-i-1}$ sur $\Sigma^*$ tels que :*

1. *$P_i U_1 \ldots U_{j-i-1} P_j$ est un facteur de $U$, et*
2. *lorsque $j > i+1$, pour chaque $\ell$ tel que $1 \leq \ell \leq j-i-1$, on a $Lev(P_{i+\ell-1}, U_\ell) = 1$.*

**Exemple 12.** *Supposons que $k = 3$. Soit le motif $P =$* AACGTGAGGTAGGTTCCATG *de longueur 20, nous le partitionnons en cinq parties de longueurs égales : $P_1 =$* AACG*, $P_2 =$* TGAG*, $P_3 =$* GTAG*, $P_4 =$* GTTC*, et $P_5 =$* CATG*. Considérons trois mots différents dont la distance de Levenshtein avec $P$ soit 3 :* AACGGAGGTAAGTTCTCATG*,* AACGTAGGCAAGTTCCATG *et* ATCGTGACGTAGGGTCCATG*. Pour chaque mot, nous montrons dans la Figure 2.7 les parties qui remplissent les conditions du Lemme 4.*

Formulons le Lemme 4 comme un pur problème de comptage afin d'établir sa preuve.

**Lemme 5.** *Soit un entier naturel $k$. Supposons que nous avons $k+2$ conteneurs numérotés de $1$ à $k+2$ ainsi que $y$ jetons avec $0 \leq y \leq k$. Il existe alors deux conteneurs $i$ et $j$, $1 \leq i < j \leq k+2$, tels que :*

1. *les conteneurs $i$ et $j$ sont vides et*
2. *pour chaque $\ell$, $i < \ell < j$, le conteneur $\ell$ contient exactement un jeton.*

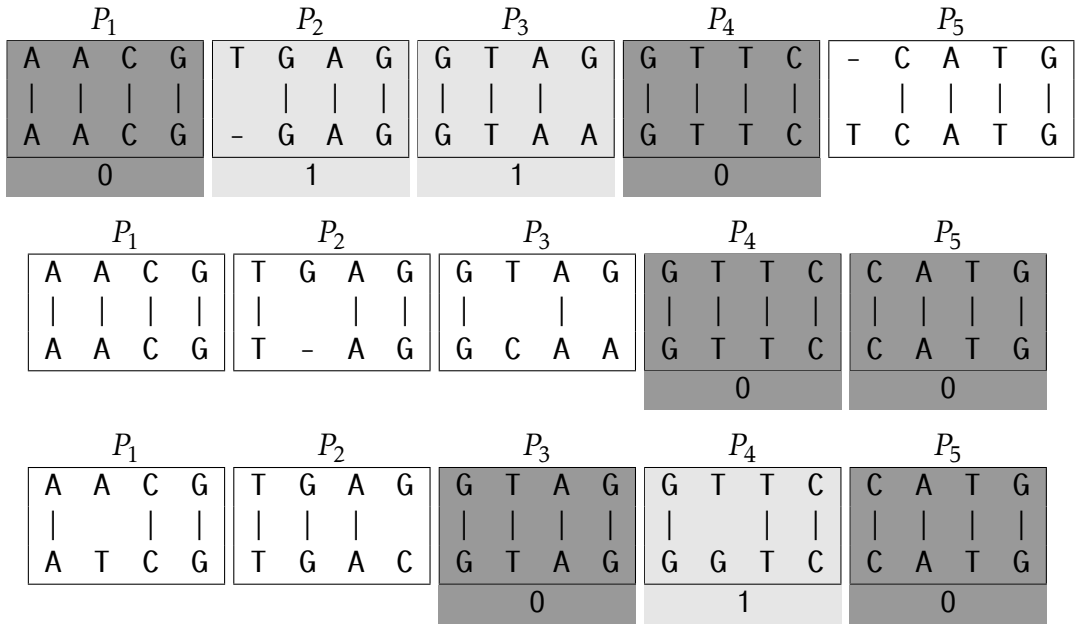



FIGURE 2.7 – Application du Lemme 4 pour les séquences de l'Exemple 12. Le motif AACGTGAGGTAGGTTCCATG est dans la partie supérieure de chaque alignement. Les deux parties sans erreur (notées $P_i$ et $P_j$ dans le lemme) possèdent une couleur de fond gris foncé, et les parties avec une erreur entourées par $P_i$ et $P_j$ ont une couleur de fond gris clair.

*Démonstration.* Soit $\mu$ le nombre de conteneurs avec au moins deux jetons, appelés des conteneurs multiples. Au total il y a au moins $\mu + 2$ conteneurs vides. Le 2 vient du fait que nous avons $k$ jetons et $k + 2$ conteneurs. Il y a alors au moins $\mu + 1$ paires de conteneurs vides $i$ et $j$ telles que chaque conteneur entre $i$ et $j$ est non vide. Les $\mu$ conteneurs multiples peuvent chacun se trouver entre une paire de conteneurs vides différente, cassant la propriété pour $\mu$ paires de conteneurs vides. Dit autrement il y a au plus $\mu$ paires de conteneurs pour lesquelles au moins un conteneur multiple est intercalé. Cela nous laisse $\mu + 1 - \mu = 1$ paire satisfaisant la propriété : tout conteneur intercalé entre une telle paire contient exactement un jeton. □

En conséquence du Lemme 4, nous pouvons mettre au point une méthode de filtrage sans perte à base de graine pour le problème de la recherche de motifs approchés avec $k$ erreurs. À cette fin nous introduisons de la terminologie qui sera utilisée dans la suite de ce document.

**Définition 15.** *Soit $P = P_1 \dots P_{k+2}$ un motif divisé en $k + 2$ parties. La* graine *01*0 pour $P$ et $k$ est l'expression régulière*

$$\cup_{i=1}^{k+1} \cup_{j=i+1}^{k+2} \; P_i \; Lev^1(P_{i+1}) \dots Lev^1(P_{j-1}) \; P_j$$

*où $Lev^1(u)$ correspond à l'ensemble des mots dont la distance de Levenshtein avec $u$ est 1. Une* sous-graine *est une expression régulière associée à une paire $(i, j)$ :*

$$P_i \; Lev^1(P_{i+1}) \dots Lev^1(P_{j-1}) \; P_j.$$

*Une* instance *de la graine, ou d'une sous-graine, est un mot $u$ de $A^*$ qui est reconnu par la graine, ou la sous-graine. Sur le texte $T$, une* occurrence *de la graine pour $P$ est un facteur de $T$ qui est une instance de la graine. Une occurrence est caractérisée par sa position de début et sa position de fin dans $T$.*

## Efficacité du filtrage

L'efficacité du filtrage est le critère principal afin d'évaluer les performances de graines. Une analyse complète du cas moyen pour les graines 01*0, comme

ce qui a été réalisé dans [13] par exemple, est en dehors du cadre de cette thèse. Avoir une formule pour le nombre attendu d'occurrences d'instances de sous-graines pour un motif donné est également non trivial puisque cette formule serait fortement dépendante de la structure du motif et de ses répétitions. Nous laissons cela ouvert pour de futures recherches. Pour l'estimer empiriquement, nous avons généré une séquence aléatoire de longueur $10^8$ sur l'ADN ainsi que 100 motifs de longueur 20. Nous avons ensuite cherché les graines 01*0 pour $k = 3$. Pour chaque motif, nous avons compté le nombre total d'occurrences des graines dans le texte, en incluant les occurrences chevauchantes. La distribution des valeurs est donnée dans la Figure 2.8. Le nombre moyen observé d'occurrences par motif est 6 665.

À titre de comparaison, nous avons également estimé l'efficacité du filtrage pour d'autres types de graines, pour le même texte et la même collection de motifs. Premièrement, nous avons considéré une graine contigüe exacte avec un filtrage sans perte, telle que décrite dans la Section 2.2.3. C'est le principe des tiroirs classique, où le motif est découpé en $k+1 = 4$ parties non chevauchantes de longueur 5. Dans la Figure 2.8, nous l'appelons la graine *partitionnement* $k+1$. Le nombre moyen d'occurrences est 385 651. Nous avons ensuite analysé la version $k+2$ du principe des tiroirs : le motif est divisé en $k+2 = 5$ parties non chevauchantes de longueur 4, et les deux parties doivent être identiques. Dans le premier cas, il faut que les deux parties exactes soient séparées par une distance compatible avec au plus $k$ erreurs : le nombre de positions entre deux parties exactes $P_i$ et $P_j$ $(i < j)$ doit aller de $p_{i+1} + \cdots + p_{j-1} - k$ à $p_{i+1} + \cdots + p_{j-1} + k$, où $p_\ell$ désigne la longueur de la partie $P_\ell$. Dans un second cas, nous avons considéré une version relâchée du Lemme 4 : toutes les parties intercalées entre les deux parties exactes doivent contenir exactement une erreur. En conséquence le nombre de positions entre $P_i$ et $P_j$ va de $p_{i+1} + \cdots + p_{j-1} - (j - i - 1)$ à $p_{i+1} + \cdots + p_{j-1} + (j - i - 1)$. Nous appelons cette graine la graine *partitionnement* $k+2$ *contraint*. Cette graine peut être vue comme une graine intermédiaire entre la graîne *partitionnement* $k+2$ et la graîne 01*0. Elle nous permet ainsi de mieux comprendre le comportement des graines 01*0. Dans ces deux cas, le nombre moyen d'occurrences est respectivement de 86 145 et 44 746 (voir Figure 2.8, *partitionnement* $k+2$ et *partitionnement* $k+2$ *contraint*). Ces graphiques montrent

que passer de $k+1$ parties à $k+2$ permet d'améliorer l'efficacité du filtrage par un facteur supérieur à 4. Les contraintes additionnelles sur les distances entre les parties permettent encore de réduire de moitié le nombre d'occurrences. Enfin, les graines 01*0 offrent un nouveau gain d'un facteur 6,7. Au final les 01*0 permettent d'avoir un filtrage 57 fois meilleure que le partitionnement classique en $k+1$ parties. Comme dernier cas, nous partitionnons le motif en trois parties seulement de longueur 6, 7 et 7. Contrairement aux quatre graines précédentes, cette graine est avec perte puisqu'elle aura des faux négatifs. Le nombre moyen d'occurrences est 36 207 (voir Figure 2.8, graine avec perte). De façon intéressante cette valeur est proche de la version avec $k+2$ parties bornées, mais cette dernière permet une sensibilité de 100%.

Ces mesures empiriques montrent que les graines 01*0 sont significativement plus sélectives que toutes les autres graines présentées en Figure 2.8. Elles offrent une efficacité de filtrage qui est de un à deux ordres de grandeur supérieure. Bien entendu, la meilleure sélectivité est obtenue au prix d'un travail additionnel pour la localisation des graines dans le texte. Chaque graine est composée de $\frac{(k+1)(k+2)}{2}$ sous-graines et certaines parties de la sous-graine devraient être cherchées avec erreurs. Cette tâche est cependant facilitée par deux propriétés des graines 01*0. Premièrement, chaque sous-graine doit avoir un préfixe et un suffixe communs avec des parties du motif et toutes les parties intercalées entre ces deux parties exactes doivent avoir exactement une erreur. Deuxièmement, les erreurs ne sont pas distribuées aléatoirement à l'intérieur de la graine, ce qui réduit drastiquement la combinatoire. Dans la suite nous allons voir la mise en pratique de ces graines 01*0.

### 2.4.2 Algorithme

Étant donné un motif $P$, nous énumérons toutes les sous-graines possibles pour le motif. Chaque sous-graine de $P$ est caractérisée par deux parties $P_i$ et $P_j$, $1 \leq i < j \leq k+2$, ayant une occurrence exacte dans le texte. D'après le Lemme 5, toutes les parties intercalées entre $P_i$ et $P_j$ doivent être cherchées avec exactement une erreur. Nous rappelons que dans le FM-index, les motifs sont cherchés de la droite vers la gauche. Nous devons donc commencer par chercher chacune

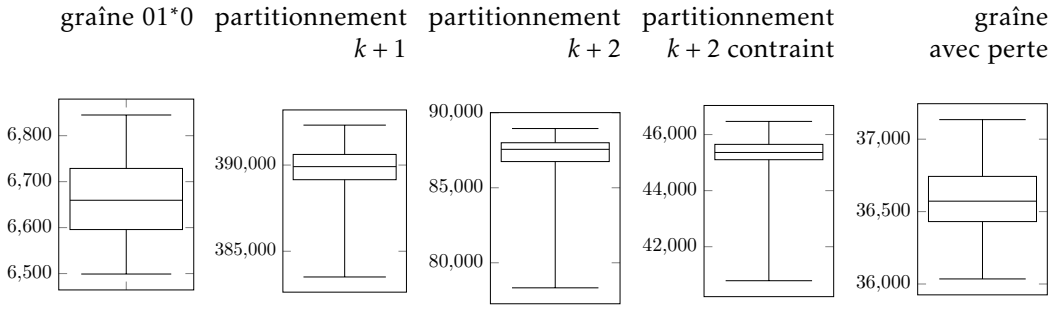
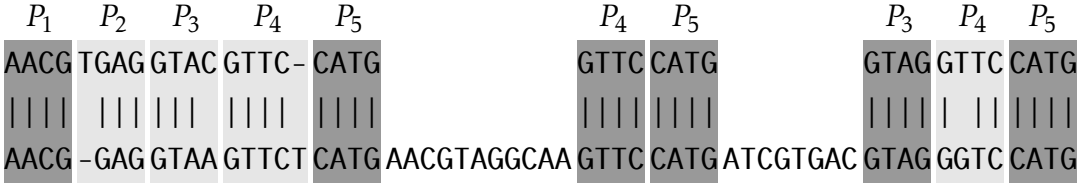



FIGURE 2.8 – Distribution du nombre d'occurrences pour cinq différentes graînes sur un jeu de données formé de 100 motifs de longueur $m = 20$ dans une séquence aléatoire de longueur $10^8$ sur de l'ADN. Les graines 01*0 et les trois types de graines par partitionnement $k+1$ ou $k+2$ sont conçues pour $k = 3$ erreurs. Ainsi pour les graines 01*0 et $k+2$, le motif est divisé en cinq parties de longueur 4. Pour le partitionnement $k+1$, le motif est divisé en 4 parties de longueur 5. Pour la graîne avec perte, chaque motif est découpé en trois parties de longueurs 6, 7 et 7 respectivement. Pour chaque boîte, le haut et le bas de la boîte sont le premier et le dernier quartiles. La ligne à l'intérieur de la boîte représente la médiane et les extrémités de la « moustache » correspondent au minimum et au maximum dans les données. Il y a en moyenne 26,85 occurrences du motif à 3 erreurs.

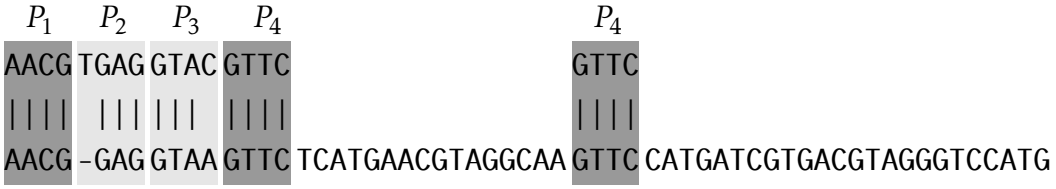
des parties $P_\ell$, avec $1 < \ell \leq k+2$, en supposant qu'il s'agit de $P_j$. Le mot exact est recherché dans l'index. Ensuite les parties précédant $P_\ell$ sont recherchées avec au plus une erreur (par backtracking comme dans BWA par exemple [97]). Lorsqu'une partie est trouvée sans erreur, on sait alors que $P_i$ a été atteint. En commençant avec $P_\ell$, il peut y avoir plusieurs parties différentes qui remplissent les critères car les parties $P_{i_1}, \ldots, P_{i_q}$ peuvent chacune apparaître de manière exacte à différentes positions dans le texte. Chacune de ces positions sera ensuite étendue pour vérifier que le motif apparaît bien à au plus $k$ erreurs. Si $P_\ell$ n'est pas trouvé sans erreur ou si une partie intercalée ne peut pas être trouvée avec au plus une erreur, la partie $P_\ell$ sera ignorée puis on passera à la valeur suivante de $\ell$. Finalement au plus $\frac{(k+1)(k+2)}{2}$ paires $(i,j)$ seront prises en compte.

**Exemple 13.** *Continuons avec l'Exemple 12, également montré dans la Figure 2.7 :* $k = 3$ *et* $P =$ AACG TGAG GTAG GTTC CATG, *qui est partitionné en 5 parties de longueur égale. Supposons que ce texte soit la concaténation de trois mots à distance 3 de* $P$ *:* $T =$*AACGGAGGTAAGTTCTCATGAACGTAGGCAAGTTCCATGATCGTGACGTAGGGTCCATG.*

– L'algorithme essaie d'abord avec $j = 5$. $P_5 =$ CATG est trouvé sans erreur dans le FM-index. Ainsi il y a une occurrence exacte dans le texte. Nous continuons à étendre la recherche dans le FM-index pour étendre $P_5$ vers la gauche et trouver toutes les valeurs possibles pour $i$. Nous trouvons $i = 4$ ($P_4$ apparaît sans erreur), $i = 3$ ($P_4$ apparaît avec une erreur et $P_3$ apparaît sans erreur) et $i = 1$ ($P_4$, $P_3$ et $P_2$ apparaissent avec une erreur et $P_1$ sans erreur). Cela nous donne trois instances différentes de graines, qui nous donnent finalement trois occurrences de graines.

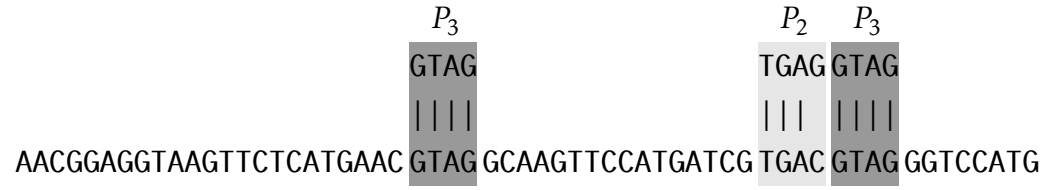



– Avec $j = 4$, GTTC apparaît sans erreur dans le FM-index, et correspond à deux occurrences dans $T$. En étendant $P_4$ vers la gauche, on ne garde qu'une instance puisque la seconde ne peut être étendue à $P_3 =$ GTAG avec au plus une erreur.




Notons que dans ce cas particulier, la première occurrence de $P$ dans $T$ est couverte par deux graînes 01*0 chevauchantes, caractérisées par $i = 1$ et $j = 5$, et $i = 1$ et $j = 4$, respectivement. Cette redondance est résolue par les phases d'extension et de vérification, qui sont décrites dans la suite.

– Avec $j = 3$, nous avons deux occurrences de GTAG dans le texte. La première ne peut pas être étendue vers la gauche avec $P_2$ = TGAG. Comme pour la seconde occurrence, $P_2$ est trouvé avec une erreur, mais $P_1$ = AACG n'apparaît pas sans erreur. L'occurrence n'est donc pas prise en compte.




– Avec $j = 2$, il y a une occurrence sans erreur de la partie TGAG dans le texte.

Désormais toutes les *instances* de graines apparaissant dans le texte sont identifiées. Nous passons alors à la phase d'élongation et de vérification.

### 2.4.3 Phases d'élongation et de vérification

Pour réaliser l'élongation d'une instance de graine, nous devons d'abord regarder la distribution des erreurs dans tout le motif, au sein de la graine et sur ses flancs. Nous savons que, par construction des graines 01*0, l'instance de la graine a une distance de Levenshtein de $j - i - 1$ avec $P_i \dots P_j$, ce qui correspond donc à $j - i - 1$ erreurs. Le lemme suivant nous permet de borner le nombre d'erreurs à la fois dans le préfixe en amont de $P_i \dots P_j$, et dans le suffixe en aval de $P_i \dots P_j$.

**Lemme 6.** *Soit $U$ un mot de $\Sigma^*$ tel que $Lev(P,U) = y \leq k$. Alors il existe une graine 01*0 $P_i \dots P_j$ telle que prefixe $P_1 \dots P_{i-1}$ contienne exactement $i - 1 - (k - y)$ erreurs et le suffixe $P_{j+1} \dots P_{k+2}$ exactement $k + 2 - j$ erreurs.*

*Démonstration.* L'hypothèse du Lemme peut être reformulée comme suit : Supposons que l'on dispose de $k+2$ conteneurs, numérotés de 1 à $k+2$, et $y$ jetons, avec $0 \leq y \leq k$. Soit $\delta = k - y$. La démonstration est par récurrence sur $k$.

Commençons par noter que si $k = 1$, les solutions pour $y = 1$ sont $(i,j) = (1,2)$ (le jeton est dans le conteneur 3), $(2,3)$ (le jeton est dans le conteneur 1) ou $(1,3)$ (le jeton est dans le conteneur 2). Si $y = 0$, l'unique solution est $(i,j) = (2,3)$.

Supposons maintenant que $k > 1$. On choisit le couple $(i,j)$ parmi tous les couples de positions possibles qui satisfont Lemme 5 et tel que $i$ soit minimal. Soit $\eta$ tel que $i - 1 - \delta + \eta$ soit le nombre de jetons présents dans les conteneurs 1 à $i-1$. Il y a trois possibilités pour $\eta$.

Si $\eta = 0$ : les jetons des conteneurs 1 à $i-1$ se montent à un total de $i-1-\delta$, et les jetons des conteneurs $j+1$ à $k+2$ à un total de $k+2-j$. Ainsi le couple de conteneurs $(i,j)$ satisfait les critères du lemme.

Si $\eta > 0$ : il y a $(i-1-\delta+\eta)+(j-i-1) = j-\delta-2+\eta$ jetons dans les conteneurs 1 à $j$. Cela laisse $y-(j-\delta-2+\eta)$ jetons dans les conteneurs $j+1$ to $k+2$. Autrement dit, nous avons $k-j+2-\eta$ jetons dans les $k-j+3$ conteneurs de $j$ à $k+2$ (comme le conteneur $j$ est vide). Par hypothèse de récurrence, cela implique qu'il existe un couple $(j', j'')$, avec $j \leq j' < j'' \leq k+2$, tel que les conteneurs $j'$ et $j''$ sont vides, les conteneurs $j$ à $j'$ contiennent $j'-j-(\eta-1)$ jetons, et les conteneurs $j''+1$ à $k+2$ contiennent $k+2-j''$ jetons. Au final, cela fait $(j-\delta-2+\eta)+(j'-j-(\eta-1)) = j'-1-\delta$ jetons pour les conteneurs 1 à $j'$. Il s'en suit que le couple $(j',j'')$ est une solution.

Si $\eta < 0$ : comme le conteneur $i$ est vide, il y a $i-1-\delta+\eta$ jetons jusqu'au conteneur $i$. Par hypothèse de récurrence, cela impliquerait que l'on peut trouver un couple $(i_1, i_2)$ qui satisfait le Lemme 5. Cela contredirait notre hypothèse initiale, puisque $i_1 < i_2 \leq i$. □

Ce lemme nous permet de ne pas chercher à étendre systématiquement toutes les instances de graines, mais uniquement celles où le nombre d'erreurs dans le préfixe précédent la graine et le suffixe suivant la graine est, en quelque sorte, bien réparti. Cela permet d'éviter des calculs inutiles quand le motif est trouvé avec plusieurs instances de graine.

Chaque instance de graine est d'abord étendue vers la gauche pour trouver $P_1 \dots P_{i-1}$ avec au plus $i-1$ erreurs. Pour gagner en efficacité, cette extension

est directement réalisée dans le FM-index, afin d'écarter le plus de candidats possibles. En effet, récupérer les positions des occurrences est la partie la plus coûteuse en temps dans le FM-index (en $O(\log^{1+\epsilon} n)$ par occurrence [43]). Une fois cette extension réalisée, toutes les occurrences du $P_1 \dots P_j$ sont récupérées, et l'extension vers la droite est réalisée dans le texte en utilisant un algorithme de programmation dynamique par bande. Le point de départ de l'extension est la position finale de l'occurrence de $P_1 \dots P_j$ dans le texte. Supposons que l'instance d'un préfixe donné $P_1 \dots P_j$ soit trouvée avec $e$ erreurs dans le FM-index. Le lemme 6 indique que $P_{j+1} \dots P_{k+2}$ peut être recherché avec au plus $k-j+2$ erreurs dans le texte. La diagonale dans l'algorithme de programmation dynamique est donc de largeur $2 \times (k-j+2)+1$.

L'extension vers la droite aurait aussi pu être réalisée dans l'index en utilisant une transformée de Burrows-Wheeler bidirectionnelle. [14, 143]. Cela augmenterait cependant l'empreinte mémoire et ne permettrait qu'un gain modeste en temps puisque de nombreux faux positifs ont été retirés lors de l'extension vers la gauche.

### 2.4.4 Implémentation

L'algorithme a été mis en œuvre dans un logiciel appelé Bwolo. Bwolo est écrit en C++, avec l'aide de la bibliothèque SeqAn et du FM-index qu'elle implante [34]. C'est un logiciel libre qui peut être téléchargé à l'adresse `http://bioinfo.lifl.fr/olo`. Dans l'implémantation, les motifs sont divisés en partie dont la longueur diffère au plus d'un caractère.

## 2.5 Performances de Bwolo

Dans cette dernière section du chapitre, nous présentons des résultats expérimentaux, qui permettent de mesurer les performances de Bwolo en termes de temps de calcul et espace mémoire.

### 2.5.1 Choix des outils pour la comparaison

Pour évaluer les performances de l'algorithme mis en œuvre dans Bwolo, nous le comparons à une sélection de logiciels existants, qui ont été choisis pour leur complémentarité.

— *Exonerate* est un outil générique pour l'alignement de séquences deux à deux, largement utilisé en bioinformatique [148]. Il utilise une méthode exacte par programmation dynamique telle que décrite en Section 2.2.2 pour réaliser la recherche. Nous l'utilisons comme une méthode de référence standard pour un algorithme exact pour notre problème.

— *RazerS3* est un programme destiné à réaliser l'alignement de lectures de séquençage de nouvelle génération [173]. Il se fonde sur le comptage de graines contigües exactes (voir sous-section 2.2.3). L'étape de vérification est réalisée grâce à une version améliorée de l'algorithme de Myers, utilisant des vecteurs de bits, proposée par Hyyrö [63]. RazerS3 fonctionne sans utiliser d'index précalculé pour le texte.

— *Bowtie2* est également un aligneur de lectures de séquençage qui utilise un filtre par facteur [89]. A la différence de RazerS3, il indexe le texte avec un FM-index, comme Bwolo. Bowtie2 a ensuite recours à du backtracking pour la gestion des erreurs et de la programmation dynamique pour obtenir l'alignement complet.

— *Readaligner* [106] est un logiciel qui met en œuvre le filtre par suffixe, décrit en Section 2.2.3. La recherche est effectuée dans un FM-index également.

— Enfin nous avons utilisé un petit outil maison pour la recherche de motifs approchés dans un FM-index à partir de la bibliothèque SeqAn. Cet outil se fonde sur un algorithme de parcours en largeur, sans filtrage préalable.

Malheureusement, nous n'avons pas pu inclure les méthodes hybrides décrites dans [137] à nos tests puisque aucun code n'est mis à disposition.

Tous ces outils ont été configurés dans le but d'atteindre la sensibilité maximale et de fournir toutes les occurrences du motif. Il s'agit donc de l'option `-exhaustive` pour Exonerate, `-filter pigeonhole -percent-identity` $[Id]$ `-recognition-rate 100` tel que $[Id] = 100\times(1-\frac{k}{m})$ pour RazerS3, `-a -L` $[Seeds]$ `-i C,`$[Seeds]$`,0` tel que $[Seeds] = \frac{m}{k+1}$ pour Bowtie2, et `--all -i 3` pour Readaligner. De plus pour chaque outil le système de score utilisé est celui permettant de calculer la distance de Levenshtein.

Les tests ont été lancés sur un seul cœur d'un serveur muni de deux Intel® Xeon® CPU E5-2420 avec 205 Go de RAM. Le temps CPU et la consommation mémoire ont été mesurés avec la commande GNU `time`.

### 2.5.2 Jeu de données 1 : séquences aléatoires

Ce premier test utilise des séquences générées aléatoirement selon une distribution uniforme sur l'ADN. La taille $n$ des séquences varie entre $10^4$ et $10^9$. Nous avons également généré aléatoirement 100 motifs de 20 nt. Nous avons mesuré le temps de calcul pour chaque outil pour $k = 2$ et $k = 3$. Les résultats sont montrés dans la Figure 2.9.

Bwolo est l'outil le plus rapide pour de longues séquences, à partir de $10^6$ nt, hormis pour $k = 2$ où Readaligner devient le plus rapide pour les séquences de longueur supérieure à $2\times10^8$. La valeur ajoutée de Bwolo est évidente avec $k = 3$. Les outils sans filtrage, Exonerate et la recherche exhaustive dans le FM-index, sont lents. Bowtie2 est lent comparé aux autres outils, en particulier avec des valeurs élevées de $n$. Cela confirme que l'heuristique de Bowtie2, qui a été conçue pour de longs motifs (au moins 50 nt) et peu d'erreurs n'est pas bien adaptée à de courts motifs avec des taux d'erreurs plus élevés. Malheureusement, il n'y a pas encore d'outil spécialisé pour ce type de problème. Dans nos tests, Bowtie2 est obligé d'utiliser une graine ayant un pouvoir filtrant faible qui laisse passer trop d'occurrences. Cela augmente de manière démesurée le temps de vérification. De manière intéressante, RazerS3, qui utilise la même graine fonctionne bien sur ces données. Cela est cohérent avec le fait qu'une méthode scannant la totalité du

texte soit dans certaines conditions, en particulier pour des valeurs élevées de $k$ et de $n$, plus efficaces que des méthodes utilisant un index [116]. Cependant Bwolo reste encore cinq fois plus rapide que RazerS3 pour des séquences de longueur $10^9$. En effet le nombre d'occurrences de graines est un ordre de grandeur inférieur dans le cas de Bwolo, ce qui diminue d'autant le temps nécessaire pour requêter le FM-index lors de la phase de vérification. Readaligner repose sur un filtrage par suffixes pour aligner les séquences. Ces filtres démontrent de bonnes performances quand la valeur de $k$ est modérée ($k = 2$ dans notre cas). Avec ce type de filtre, la configuration la moins avantageuse est quand toutes les parts contiennent exactement une erreur, hormis la dernière part qui est exacte. Quand $k = 3$, cette dernière part fait 5nt de long. Chercher un tel motif conduit à de nombreux faux positifs.

Pour $k = 2$, nous observons qu'il y a de moins en moins de différence en temps de calcul entre FM-index et Bwolo à mesure que la séquence est plus longue. Le premier prend 18,4 s pour une séquence de 1 Go alors que le dernier prend 13,8 s. Cette différence minime est en fait trompeuse. Le chargement de l'index depuis le disque (index qui est le même dans les deux cas) et la désérialisation des structures de données prend 12 s sur cette séquence-là. Si nous ignorons ce temps de chargement, la méthode à base de graînes 01*0 est alors trois fois plus rapide que l'approche par exploration en largeur. Avec un taux d'erreur plus élevé ($k = 3$) Bwolo est 75 fois plus rapide que FM-index. Puisque nous avons mentionné le temps de chargement du FM-index, notons que Readaligner met 1 s à désérialiser le FM-index, et RazerS3 8 s à charger une séquence de 1 Go depuis le disque. Concernant Readligner, cela explique également pourquoi c'est l'outil le plus rapide pour de longues séquences quand $k = 2$ : le gain vient du chargement de l'index, qui est plus efficace qu'avec RazerS3 ou Bwolo.

Tous les outils ont une consommation mémoire raisonnable, indépendante de la valeur de $k$ et qui croît linéairement avec la taille du texte. Par exemple, elle est de 27 Mo pour Bwolo, 99 Mo pour Bowtie2, 25 Mo pour RazerS3, 13 Mo for Readaligner, et 31 Mo pour Exonerate pour $n = 10^7$. La consommation mémoire de readaligner, Bwolo et Bowtie2 est principalement due à la taille du FM-index. Elle est plus importante pour Bowtie2 car le logiciel indexe également le renversé du texte. Il est assez surprenant que RazerS3 et Exonerate aient un

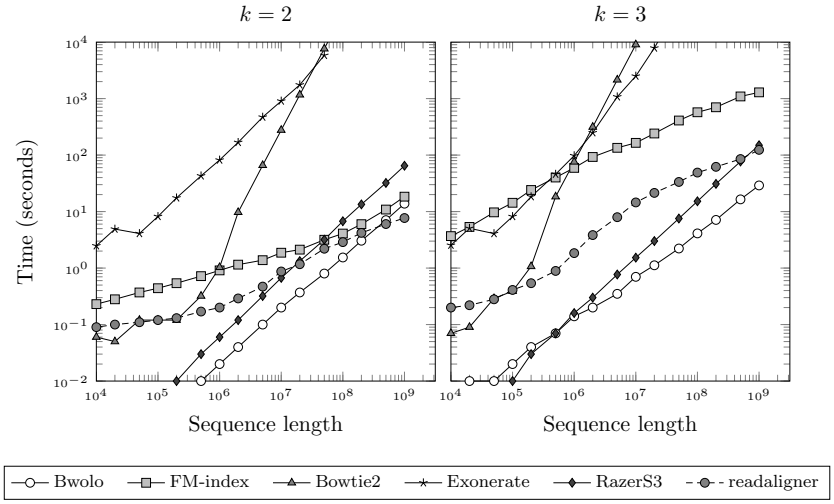



FIGURE 2.9 – Temps d'exécution sur 100 motifs générés aléatoirement. Les deux axes ont une échelle logarithmique. Bwolo correspond à l'algorithme présenté précédemment, FM-index fait référence à la recherche par exploration en largeur en utilisant un FM-index.

pic mémoire du même ordre de grandeur que les logiciels précédents. Il est possible qu'ils chargent tous les deux le texte en mémoire et qu'ils y conservent les résultats.

### 2.5.3 Jeu de données 2 : lectures de séquençage

| | construction de l'index | | 10 000 lectures | | $10^7$ lectures | |
|---|---|---|---|---|---|---|
| | temps | mémoire | temps | mémoire | temps | mémoire |
| Bwolo | 7 594 | 9 584 | 97 | 6 522 | 55 493 | 9 054 |
| RazerS3 | 0 | 0 | 502 | 6 469 | 467 413 | 152 045 |
| Readaligner | 26 060 | 5 500 | 2 460 | 3 600 | ND | ND |
| Bowtie2 | 10 584 | 5 379 | 156 164 | 8 260 | ND | ND |

TABLEAU 2.1 – Temps d'exécution sur le jeu de test du génome humain. Les temps sont en secondes et la mémoire en Mo. ND : non déterminé.

Afin de tester Bwolo sur des données externes, nous avons utilisé le benchmark de SCHBATH et al. [142], qui est constitué de courtes lectures de séquençages

simulées. Le jeu de données $\mathcal{H}_3$ contient 10 millions de lectures de 40 nt qui ont été générées à partir du génome humain (assemblage 37.1 du NCBI, 25 chromosomes totalisant 2,7 Gbp) avec exactement trois substitutions. Par rapport au test précédent, cela nous permet d'évaluer la performance du logiciel avec des motifs plus longs, et donc de plus grandes graînes. Le nombre maximal d'erreurs $k$ est de 3 (en incluant les insertions et délétions, pas seulement des substitutions). Nous avons lancé Bwolo, RazerS3 ainsi que Bowtie2 sur la totalité du jeu de lectures ($10^7$ lectures). Puisque nous n'avons pas été en mesure d'obtenir de résultats avec Bowtie2 et Readligner sur le jeu de données complet dans un laps de temps raisonnable, nous avons également utilisé un échantillon aléatoire de 10 000 lectures. Les lectures sont présentées dans le Tableau 2.1. À l'image du test précédent, Bwolo réalise les meilleures performances. La différence entre Bwolo et les autres outils est encore plus frappante que dans le test précédent : cela est dû au temps de chargement de l'index. Il devient négligeable pour ce jeu de données alors qu'il constituait une partie importante du temps de recherche avec un jeu de données plus petit tel que dans le test précédent.

Chapitre 3

# Application à la recherche de cibles de miARN chez *A. thaliana* et *A. lyrata*

Nous avons vu dans le chapitre 1 de ce document que les miARN sont des régulateurs importants de l'expression des gènes qui agissent au travers de l'identité de séquence avec leurs gènes ciblés. Les réseaux de régulation qui en résultent ont une importance centrale dans de nombreux mécanismes biologiques et sont impliqués dans de très nombreuses fonctions biologiques testées à ce jour. Cela inclut en particulier des processus du développement et d'adaptation [75, 157], ou l'implication dans de nombreuses pathologies humaines [39]. Cependant, la structure générale des réseaux de régulation contrôlés par les miARN reste mal connue. Ceci est dû en particulier à un défi expérimental pour identifier les propriétés générales des cibles fonctionnelles doublé d'un défi algorithmique pour rechercher les cibles de miARN à l'échelle d'un génome. Ce second verrou est celui que j'ai cherché à lever avec le développement de Bwolo, présenté dans le chapitre 2 de ce document.

Ce troisième chapitre présente une application de Bwolo à l'analyse de deux génomes de plantes, *A. thaliana* et *A. lyrata*. Pour ces deux espèces, le génome a été séquencé et annoté de façon relativement complète et des études de séquençage massif de petits ARN ont permis de prédire la présence d'un ensemble de miARN. La question que nous abordons ici est celle de la répartition des cibles des miARN le long du génome. En effet, nous avons dit en section 1.2.2 que le mode d'action préférentiel pour les miARN de plantes implique un ciblage des parties transcrites des gènes. Des études récentes ont toutefois mis en évidence une diversité de mécanismes inattendue, avec par exemple des cibles dans les introns des gènes [112], des cibles régulées par méthylation [177] ou des cibles situées en amont des gènes [36, 159]. Nous proposons dans ce chapitre une étude exhaustive de la distribution des cibles des miARN connus de *A. thaliana* et *A. lyrata* dans les exons et les régions intergéniques. En raison de son efficacité en temps de calcul et du peu d'hypothèses faites sur le modèle d'interaction entre le miARN et sa cible, Bwolo est un outil adapté à une telle recherche à grande échelle de sites potentiellement non-canoniques. L'efficacité de Bwolo permet également de mettre en œuvre des approches par permutation, permettant de tester un ensemble d'hypothèses nulles.

## 3.1 Identification des cibles potentielles

### 3.1.1 Préparation des données

Nous avons utilisé la version annotée du génome d'*A. thaliana* (version TAIR9 [87]) et d'*A. lyrata* (v1.0, [61] avec l'annotation récemment mise à jour par Rawat et al. [129]). Les ADNc d'*A. thaliana* de TAIR9 ont été également téléchargés. Chacune de ces séquences a été convertie en une séquence sur l'alphabet {ACTGN} afin d'éliminer les quelques symboles d'ambigüité du système IUPAC pour les remplacer par des N.

L'ensemble des miARN connus des deux espèces a été téléchargé à partir de la base de données miRBase 21 [78]. Cela consiste en 427 et 384 miARN distinct chez *A. thaliana* et chez *A. lyrata* respectivement. Devant la difficulté à distinguer les motifs de miARN réellement actifs des erreurs d'annotation, miRBase contient un sous-ensemble de 135 et 56 miARN désignés comme «*fiables*» (*highly confident*) chez chacune des deux espèces, sur la base d'un nombre important de lectures de séquences (au moins 10 reads sur chaque bras de la structure tige-boucle ou 5 reads sur chaque bras du précurseur avec plus de 100 reads au total). Cette catégorie est de ce fait principalement identifiée sur la base de leur niveau d'expression qui dépend de la profondeur des données de séquençage et n'est pas en tant que telle un garant de l'importance de leur rôle fonctionnel.

Certains précurseurs de miARN produisent plus d'un miARN mature, avec par exemple le miARN et le miARN*, ou des miARN chevauchants mais distincts. Ces miARN ont des séquences différentes et donc des cibles potentiellement différentes. De ce fait, nous les avons considérés séparément dès lors qu'ils étaient annotés comme des miARN matures dans miRBase. De la même façon nous avons conservé séparément les miARN appartenant à une même famille dès lors qu'ils étaient produits dans des locus différents. Nous avons retiré un seul miARN annoté dans le génome d'*A. lyrata* (Aly-miR848-5p) en raison de sa taille exceptionnellement petite (seulement 17 nucléotides).

### 3.1.2 Construction de miARN aléatoires

À partir des miARN extraits de miRBase, nous avons généré des séquences synthétiques, afin de pouvoir comparer nos observations sur les cibles avec une distribution attendue sous un ensemble d'hypothèses nulles. Pour cela, nous avons utilisé quatre méthodes de génération de séquences aléatoires, afin de réaliser pour chaque ensemble de miARN considéré 1 000 ensembles ayant la même taille et la même distribution de tailles de séquences que l'ensemble de départ.

— *Génération aléatoire* : ce sont des séquences générées à partir d'un tirage équiprobable entre les quatre nucléotides {ACGT} («génération aléatoire»).

— *Mélange nucléotide local* : de façon à préserver la composition *nucléotidique* des miARN réellement produits par le génome considéré, nous avons généré des séquences de pseudo-miARN issues de la permutation aléatoire des nucléotides de chaque miARN, de la même manière que Rhoades et al. [135].

— *Mélange dinucléotide local* : comme l'énergie d'hybridation d'un miARN à sa cible suit un modèle des plus proches voisins [160], nous avons réalisé des permutations garantissant de conserver la composition en *dinucléotides*, comme cela est fait habituellement dans le contexte de la prédiction des structures d'interaction impliquant des acides nucléiques [11, 165, 176]. La génération de telles permutations a été faite suivant l'algorithme de Altschul et Erickson [5], pour chaque séquence de miARN individuellement. Dans ce cas, le nombre de 1000 séquences est à prendre avec précaution, car l'échantillonnage complet est limité.

— *Mélange dinucléotide global* : nous avons appliqué le même type de mélange que dans le cas précédent, en considérant l'ensemble des miARN du jeu utilisé, et non plus chaque miARN individuellement. Contrairement au mélange dinucléotide local, le nombre de permutations possibles n'est pas contraint par la taille de chaque séquences. Cette distinction entre mélanges dinucléotidiques local et global permet de déterminer si les éventuels écarts observés résultent de propriétés générales des miARN

dans leur ensemble, ou sont dus à une composition dinucléotidique particulière de certains miARN.

### 3.1.3 Prédiction des cibles

Afin d'évaluer les performances de Bwolo sur des données biologiques réelles, nous nous sommes concentrés sur le génome d'*A. thaliana*, qui est à la fois le mieux annoté et dans lequel l'effort de validation fonctionnel du rôle des miARN a été le plus important. Nous avons utilisé une base de données de 331 interactions miARN-cible expérimentalement validées réalisée par Srivastava et al. [151] et avons déterminé la proportion de ces interactions validées que Bwolo était effectivement capable d'identifier (valeur de rappel) et à l'inverse quelle proportion de toutes les interactions prédites par Bwolo appartenaient à l'ensemble des interactions validées (valeur de précision). Un outil de prédiction idéal aurait un rappel de 100% et une précision de 100%.

Nous avons utilisé Bwolo pour rechercher de manière exhaustive dans les génomes sur les deux brins les différents ensembles de miARN associés en autorisant au plus trois erreurs, tel que cela a été décrit dans le chapitre 2 (voir section 2.4). Nous avons considéré comme *cibles potentielles d'un miARN* l'ensemble des séquences qui lui sont fortement similaires. Par complémentarité, la cible se trouve sur le brin opposé. Pour l'analyse de la distribution des cibles à travers le génome, nous avons à ce stade de l'étude ignoré les cibles chevauchant un site d'épissage. La taille moyenne d'un exon chez *A. thaliana* est égale à 488 nucléotides [8], la probabilité d'être sur une fenêtre chevauchante de longueur 21nt est faible, ce qui ne devrait pas affecter grandement nos résultats.

Nous avons ensuite calculé une p-valeur correspondant à la proportion de permutations dans lesquelles l'hypothèse nulle considérée est vrai.

Cependant dans le cadre spécifique de la comparaison de valeur de rappel et de précision, afin que les résultats soient le plus comparable possible nous avons prédit les cibles sur les ADNc plutôt que sur les séquences génomiques.

| | post-filtrage | Valeur | p-valeur | | | |
|---|---|---|---|---|---|---|
| | | | GA | SNL | DNL | DNG |
| rappel | sans | 48,0% | <0,001 | <0,001 | <0,001 | <0,001 |
| | avec | 42,9% | <0,001 | <0,001 | <0,001 | <0,001 |
| précision | sans | 13,6% | <0,001 | <0,001 | <0,001 | <0,001 |
| | avec | 16,7% | <0,001 | <0,001 | <0,001 | <0,001 |

Tableau 3.1 – Valeurs de rappel et de précision des cibles identifiées par Bwolo avec ou sans post-filtrage, l'ensemble de comparaison était constitué de 331 interactions miARN-cibles expérimentalement validées. Le post-filtrage est constitué d'un ensemble de critères d'alignement pour la recherche de cibles de miARN [95]. Les p-valeurs sont obtenues par différentes procédures de génération aléatoire des séquences de miARN selon l'hypothèse nulle « le rappel et la précision ne sont pas supérieurs à l'attendu aléatoire ». GA : génération aléatoire, SNL : mélange simple-nucléotide local, DNL : mélange dinucléotide local , DNG : mélange dinucléotide global.

## 3.2 Rappel et précision pour les cibles canoniques de *A. thaliana*

Concernant le génome de *A. thaliana*, nous avons pu utiliser une base de données de 331 interactions miARN-cible expérimentalement validées réalisée par Srivastava et al. [151]. Ce sont des cibles dites *canoniques*, dans le sens où elles sont localisées dans un exon d'un gène.

Dans le cadre spécifique de cette étude nous avons prédit les cibles de miARN non sur le génome mais sur l'ensemble des ADNc ceci dans un soucis de comparaison avec Srivastava et al. [151]. De plus, nous avons limité la recherche de cibles aux seuls miARN qui possède au moins une cible validée expérimentalement (70 miARN). Nous obtenons ainsi 1 214 prédictions d'interactions uniques cible-miARN pour tous les miARN, dont 159 correspondent à l'une des 331 interactions du jeu de données. Cela correspond à un rappel de 48,0% et une précision de 13,1%.

Comme Bwolo utilise un modèle très simple pour la recherche de cibles, nous avons cherché à évaluer l'effet de l'application d'un filtre basé sur des règles empiriques, suivant les recommandations issues d'études sur les cibles

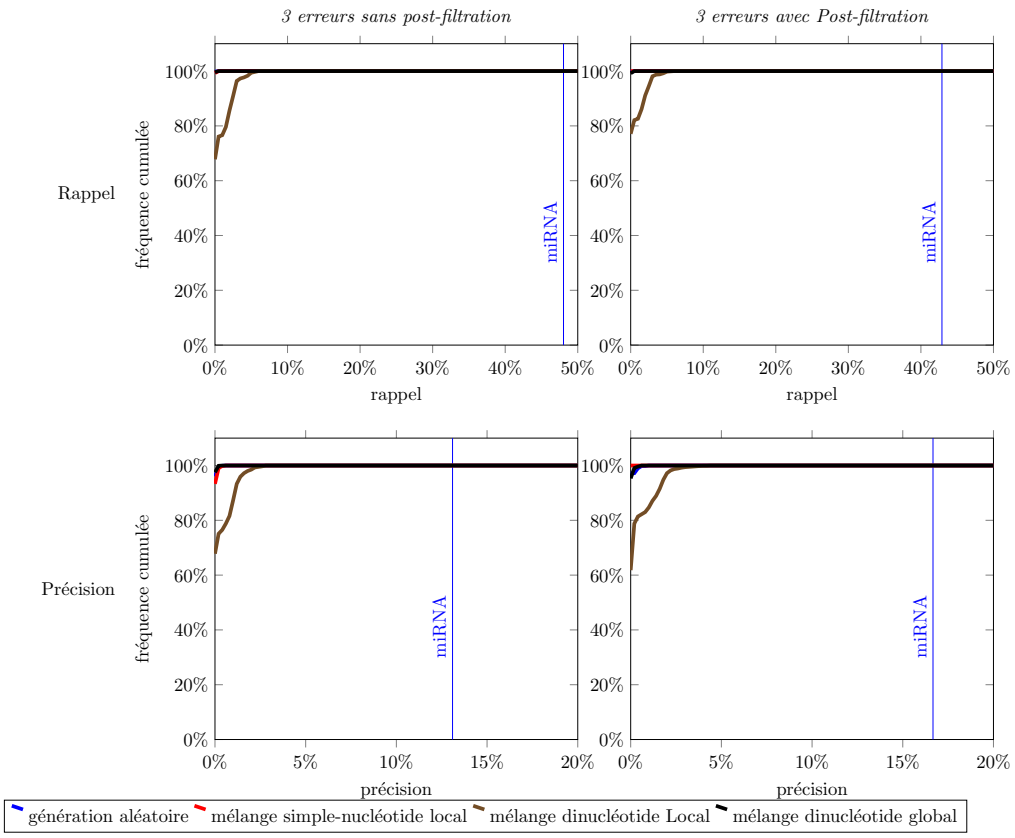



FIGURE 3.1 – Analyse du rappel (en haut) et de la précision (en bas) de la prédiction de cibles de miARN dans les ADNc en comparaison à un ensemble de 331 interactions cibles-miARN expérimentalement vérifiées dans les ARNm d'*A. thaliana* [151] en sortie brute de Bwolo (à gauche) ou après filtrage par un ensemble de règles empiriques de conditions d'alignement (à droite, voir sous-section 3.1.3). La valeur observée pour les miARN est représentée par une barre verticale sur chaque graphe et la fréquence cumulée attendue sous différentes hypothèses nulles (voir sous-section 3.1.2) est représentée par des courbes de couleurs.

de miARN précédemment validées dans le génome d'*Arabidopsis* Schwab et al. [144] et Allen et al. [4] (voir sous-section 1.3.2). Ces règles s'appliquent pour des cibles de miARN canoniques, tel que c'est le cas ici. Elles sont mises en œuvre dans le logiciel SoMART [95], par exemple. Celles-ci sont :

- pas plus de deux erreurs consécutives dans le duplex miARN/cible,
- pas d'erreurs adjacentes entre les positions 2 et 12 du duplex miARN/cible (par rapport à l'extrémité 5' du miARN),
- pas d'erreur aux positions 10 et 11,
- un score d'erreur inférieur ou égal à 2,5 entre les positions 1 à 12 du duplex miARN/cible, toutes erreurs comptant pour 1 sauf les wooble G-U qui comptent pour 0,5,
- l'énergie libre minimale (MFE pour *minimal free energy*) du duplex miARN/-cible doit être > 0.74 du MFE d'un match parfait pour ce miARN. Le calcul de ce ratio est réalisé à l'aide d'une approximation en comptant le nombre de ponts hydrogène impliqués dans le duplex, une liaison C-G fournissant 3 ponts, A-U et G-U en fournissant seulement 2.

Contrairement à la plupart des outils existants qui n'appliquent ces critères qu'au seul appariement optimal donné par l'algorithme d'alignement classique (typiquement dérivé de l'algorithme de Smith et Waterman), ici, j'ai appliqué ces critères à tous les appariements possibles entre chaque miARN et chacune de ses cibles potentielles, incluant donc les alignements considérés comme suboptimaux mais conservés dès qu'ils correspondent aux critères définis. Enfin, à cause de l'existence de cibles fortement chevauchantes dues aux différentes possibilités d'aligner deux séquences en introduisant des erreurs ou des gaps, nous n'avons considéré que la cible la plus centrale à chaque fois que plusieurs cibles pour un même miARN se chevauchaient sur un même brin.

Ce filtre supplémentaire sur les prédictions réduit le nombre de cibles prédites de 1 214 à 852, dont 142 correspondent aux interactions connues, contre 159 sans filtre. Cela correspond à une diminution légère du rappel à 42,9% (soit un baisse d'environ 10% par rapport à la valeur sans post-filtrage) mais permet une augmentation sensible de la précision à 16,6% (soit un gain de précision d'environ 20% par rapport à la valeur sans post-filtrage). Bien que ces valeurs puissent sembler relativement faibles en regard d'autres outils présentés par

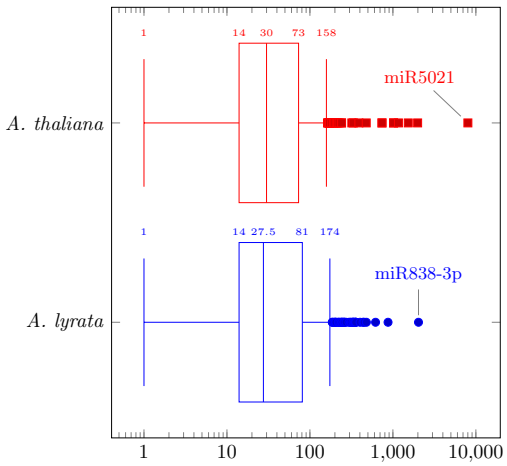



FIGURE 3.2 – Distribution du nombre de cibles potentielles pour tous les miARN dans le génome complet d'*A. thaliana* (en haut) et d'*A. lyrata* (en bas). La barre centrale est la médiane, le rectangle central représente les deux quartiles. Sous l'hypothèse d'une distribution normale, les deux barres verticales couvrent 98% des données. Les valeurs extrêmes sont représentées individuellement.

SRIVASTAVA et al. [151], elles sont dans tous les cas bien au-delà des valeurs attendues aléatoires, quelle que soit la procédure de permutation utilisée pour générer l'hypothèse nulle ($p - value < 0.001$ Tableau 3.1 et Figure 3.1).

## 3.3 Analyse à l'échelle des génomes de *A. thaliana* et *A. lyrata*

### 3.3.1 Nombre de cibles potentielles de miARN

Nous avons ensuite analysé l'intégralité des cibles potentielles de notre ensemble de miARN, dans tout le génome de *A. thaliana* et *A. lyrata*, sans se restreindre aux seuls ARNm et en intégrant tous les miARN qu'ils aient fait l'objet d'une interaction miARN-cible fonctionnellement validée ou non. Ne connaissant pas les propriétés de reconnaissance de ces cibles potentiellement non canoniques par les miARN, nous choisissons en conséquence de n'utiliser

| | miARN | Nb de cibles par miARN | p-valeur | | | |
|---|---|---|---|---|---|---|
| | | | GA | SNL | DNL | DNG |
| *A. thaliana* | tous | 87,83 | <0,001 | <0,001 | <0,001 | <0,001 |
| | fiables | 39,56 | <0,001 | <0,001 | <0,001 | <0,001 |
| *A. lyrata* | tous | 71,30 | <0,001 | <0,001 | 0,005 | 0,002 |
| | fiables | 48,64 | 0,009 | <0,001 | 0,089 | 0,098 |

Tableau 3.2 – Comparaison des nombres moyens de cibles par miARN dans le génome d'*A. thaliana* et *A. lyrata* pour l'ensemble des miARN (tous) et l'ensemble restreint des miARN «fiables» (fiables). Les p-valeurs sont obtenues par différentes procédures de génération aléatoire des séquences selon l'hypothèse nulle «le nombre moyen de cibles par miARN n'est pas supérieur à l'attendu aléatoire». : GA : génération aléatoire, SNL : mélange simple-nucléotide local, DNL : mélange dinucléotide local , DNG : mélange dinucléotide global.

que les résultats bruts de Bwolo, sans faire appel à la procédure de post-filtrage, spécifiquement adaptée à la recherche de cibles canoniques chez *A. thaliana*. Les résultats sont montrés en Figure 3.2.

La majorité des miARN possède entre 14 et 73 cibles potentielles de miARN chez *A. thaliana* et entre 14 et 81 cibles potentielles de miARN chez *A. lyrata*, pour une médiane de 30 cibles pour *A. thaliana* et 27,5 pour *A. lyrata*. Ainsi, malgré un génome 40% plus petit chez *A. thaliana*, il y a à peu près le même nombre de cibles potentielles de miARN chez *A. thaliana* que chez *A. lyrata*. Dans les deux cas, la distribution du nombre de cibles est marquée par l'existence d'un petit nombre de miARN possédant un nombre de cibles particulièrement élevé. En particulier, miR5021 chez *A. thaliana* possède à lui seul 8 013 cibles et miR838-3p chez *A. lyrata*, 2 035 cibles (Figure 3.2). Les séquences de ces deux miARN sont `UGAGAAGAAGAAGAAGAAAA` et `UUUUCUUCUUCUUCUUGCACA`. Elles sont pratiquement le complémentaire inversé l'une de l'autre (3 erreurs) et se distinguent par 5 répétitions de 3 nucléotides (`GAA` pour miR5021 et `UUC` pour miR838-3p).

Le nombre moyen de cibles par miARN est bien inférieur pour le sous-ensemble restreint de miARNs «*fiables*» en comparaison à l'ensemble de miARN dans les deux espèces avec en moyenne 39,56 cibles par miARN «fiables» contre 87,83 cibles par miARN chez *A. thaliana* et 48,64 cibles par miARN «fiables»

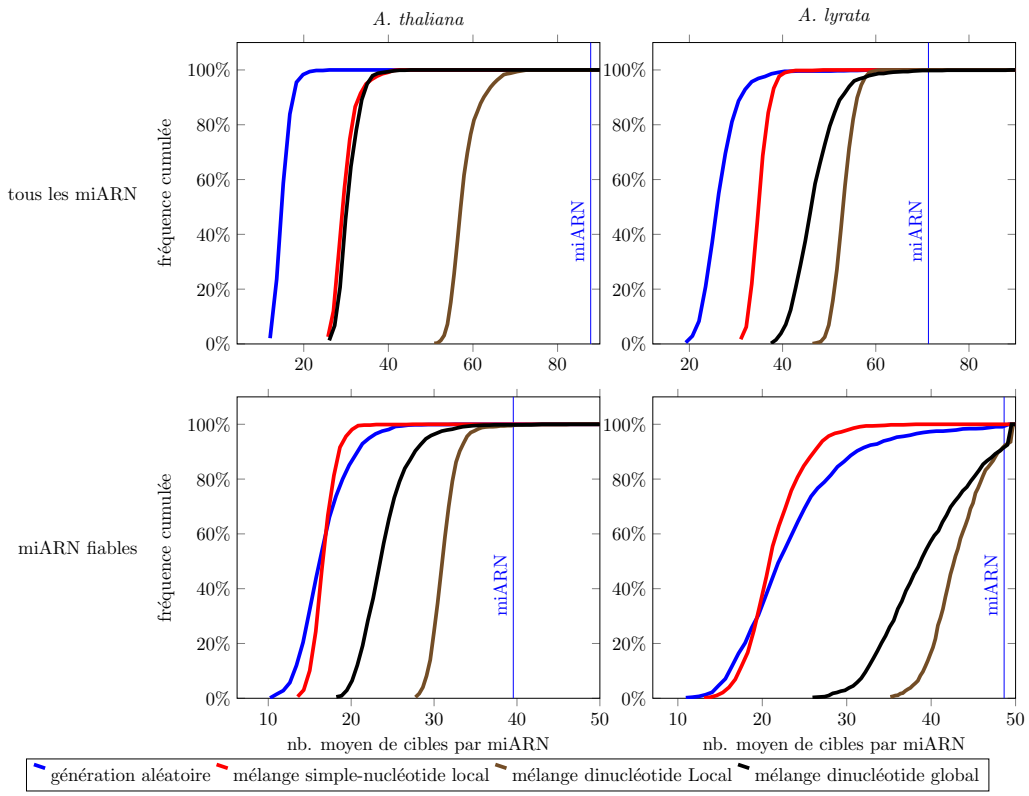



FIGURE 3.3 – Comparaison du nombre moyen de cibles potentielles par miARN par rapport à l'aléatoire attendues dans le génome d'*A. thaliana* (gauche) et *A. lyrata* (à droite) pour l'ensemble des miARN (en haut) et l'ensemble restreint des miARN «fiables». La valeur observée est représentée par une barre verticale sur chaque graphe et la fréquence cumulée attendue sous différentes hypothèses nulles (voir sous-section 3.1.2) est représentée par des courbes de couleur.

| | miARN | Nb de cibles par miARN | p-valeur | | | |
|---|---|---|---|---|---|---|
| | | | GA | SNL | DNL | DNG |
| *A. thaliana* | tous | 17,19 | <0,001 | <0,001 | <0,001 | <0,001 |
| | fiables | 9,07 | <0,001 | <0,001 | 0,007 | 0,002 |
| *A. lyrata* | tous | 5,34 | <0,001 | <0,001 | <0,001 | 0,001 |
| | fiables | 3,96 | 0,001 | <0,001 | 0,087 | 0,017 |

Tableau 3.3 – Comparaison du nombre moyen de cibles par miARN localisées dans un exon chez *A. thaliana* et *A. lyrata* pour l'ensemble des miARN (tous) et l'ensemble restreint des miARN «fiables» (fiables). Les p-valeurs sont obtenues par différentes procédures de génération aléatoire des séquences selon l'hypothèse nulle «le nombre moyen de cibles potentielles par miARN dans un exon n'est pas supérieur à l'attendu aléatoire». GA : génération aléatoire, SNL : mélange simple-nucléotide local, DNL : mélange dinucléotide local , DNG : mélange dinucléotide global.

contre 71,30 cibles par miARN chez *A. lyrata* (Tableau 3.2). Nous avons comparé ce nombre de cibles à ce qui serait attendu avec des séquences aléatoires, suivant les quatre modèles présentés en Section 3.1.2. Les résultats sont présentés dans le Tableau 3.2. De façon frappante, le nombre observé moyen de cibles potentielles est très largement supérieur à son attendu aléatoire. Ceci est vrai quelle que soit la procédure de permutation utilisée ($p < 0.001$), à l'exception des permutations dinucléotides globales ou locales chez *A. lyrata* lorsque seul le sous-ensemble restreint de miARN dits *fiables* est considéré ($p = 0.089$ et $p = 0.098$ respectivement Figure 3.3). Il est possible que ce résultat négatif soit dû au nombre faible de miARN «fiables» annotés dans le génome dû à l'effort d'annotation plus limité chez cette espèce (seulement 56). Les génomes d'*A. thaliana* et d'*A. lyrata* sont donc globalement enrichis en cibles potentielles de miARN.

### 3.3.2 Distribution des cibles potentielles de miARN

Pour comprendre d'où peut venir l'enrichissement global en cibles de miARN, nous distinguons entre cibles canoniques, localisées dans les exons, et cibles non canoniques situées en dehors des exons. Si l'ensemble des cibles prédites

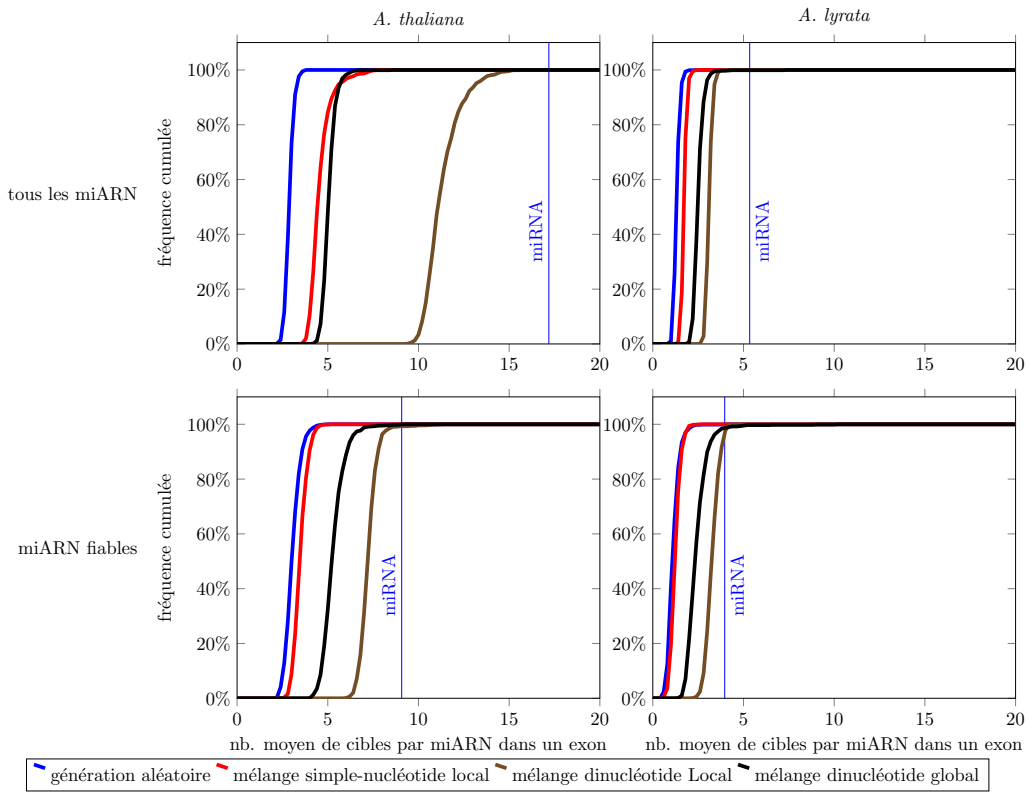



FIGURE 3.4 – Comparaison du nombre moyen des cibles potentielles et attendues par miARN localisées dans un exon chez *A. thaliana* (gauche) et *A. lyrata* (à droite) pour l'ensemble des miARN (en haut) et l'ensemble restreint des miARN «fiables». La valeur observée est représentée par une barre verticale sur chaque graphe et la fréquence cumulée attendue sous différentes hypothèses nulles (voir sous-section 3.1.2) est représentée par des courbes de couleur.

de miARN correspondait à des cibles canoniques, alors elles devraient toutes se situer dans des exons. Quand on restreint l'analyse aux seules régions exoniques, il apparait que celles-ci sont en effet fortement enrichies en cibles potentielles (voir Tableau 3.3). Ainsi, chez *A. thaliana*, un miARN a en moyenne 17,19 cibles dans un exon pour tous les miARN et ce nombre diminue à 9,07 pour le seul sous-ensemble des miARN fiables. Chez *A. lyrata*, un miARN a en moyenne 5,34 cibles dans un exon et ce nombre diminue à 3,96 pour le seul sous-ensemble des miARN fiables. Ces valeurs sont également plus importantes que l'attendu aléatoire pour la majorité des cas ($p < 0.001$). Seuls les miARN fiables d'*A. lyrata* ne sont que marginalement significatifs par rapport au mélange dinucléotidique local ($p = 0.087$). Les régions géniques sont donc également enrichies en cibles potentielles de miARN (Figure 3.4 Tableau 3.3).

Nous avons alors voulu savoir si l'enrichissement dans les exons était proportionnellement différent de l'enrichissement global, et avons déterminé la proportion de cibles présentes dans un exon par rapport à l'ensemble de toutes les cibles du génome. Chez *A. thaliana*, il y a 19,3% de cibles dans un exon pour tous les miARN et 22,0% pour les miARN fiables (Tableau 3.4). De façon surprenante, ces proportions sont plus faibles que l'attendu aléatoire (Figure 3.5), ce qui implique que l'enrichissement en cibles est proportionnellement plus important en dehors des exons que dans les exons eux-mêmes. Le signal semble être différent chez *A. lyrata*, 8,3% des cibles sont dans un exon pour tous les miARN et 8,4% pour les miARN fiables. Cependant, contrairement à *A. thaliana*, ces valeurs tendent à être légèrement supérieures et non inférieures à leur attendu aléatoire pour l'ensemble des miARN, indiquant que chez cette espèce, les exons semblent bien enrichis en cibles, mais ne s'en distinguent pas pour le sous-ensemble des miARN fiables (Figure 3.5 et Tableau 3.4).

### 3.3.3 Distribution des cibles intergéniques

L'utilisation de Bwolo nous a permis de rechercher dans le génome complet des cibles potentielles, et en particulier dans les régions intergéniques. Ces résultats précédents indiquent que si les régions exoniques sont bien enrichies en cibles, la proportion de cibles exoniques est en fait plus faible qu'attendu

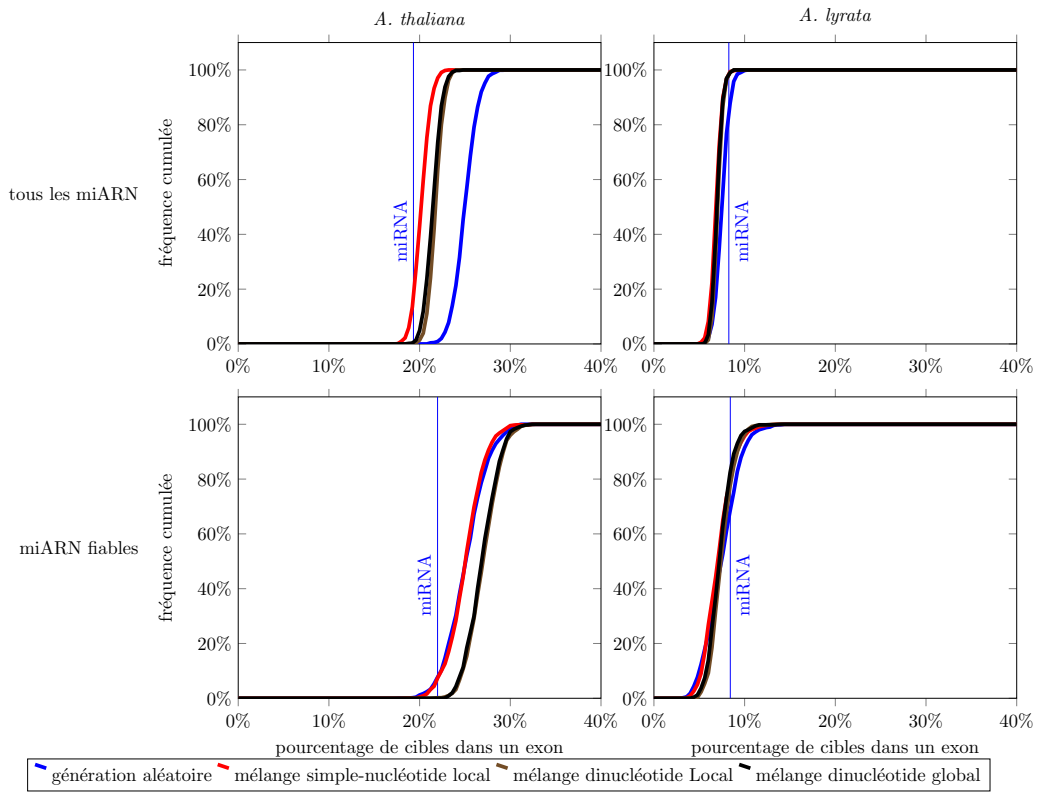



FIGURE 3.5 – Comparaison du pourcentage moyen des cibles potentielles et attendues localisées dans un exon chez *A. thaliana* (gauche) et *A. lyrata* (à droite) pour l'ensemble des miARN (en haut) et l'ensemble restreint des miARN «fiables». La valeur observée est représentée par une barre verticale sur chaque graphe et la fréquence cumulée attendue sous différentes hypothèses nulles (voir sous-section 3.1.2) est représentée par des courbes de couleur.

| | miARN | % de cibles exoniques | p-valeur | | | |
|---|---|---|---|---|---|---|
| | | | GA | SNL | DNL | DNG |
| *A. thaliana* | tous | 19,32% | >0,999 | 0,920 | >0,999 | 0,998 |
| | fiables | 21,98% | 0,948 | 0,957 | 0,999 | >0,999 |
| *A. lyrata* | tous | 8,26% | 0,302 | 0,045 | 0,061 | 0,044 |
| | fiables | 8,42% | 0,386 | 0,266 | 0,314 | 0,263 |

Tableau 3.4 – Comparaison du pourcentage moyen des cibles localisées dans un exon chez *A. thaliana* et *A. lyrata* pour l'ensemble des miARN (tous) et l'ensemble restreint des miARN «fiables» (fiables). Les p-valeurs sont obtenues par différentes procédures de génération aléatoire des séquences selon l'hypothèse nulle «la proportion de cibles dans un exon n'est pas supérieure à l'attendu aléatoire». GA : génération aléatoire, SNL : mélange simple-nucléotide local, DNL : mélange dinucléotide local , DNG : mélange dinucléotide global.

par hasard chez *A. thaliana*, suggérant qu'un grand nombre de cibles de miARN pourrait être intergénique. Nous avons cherché à en savoir plus sur ces régions.

Dans un premier temps, nous avons voulu savoir si ces cibles potentielles de miARN correspondent à des régions non codantes conservées (*conserved non coding sequences* ou CNS) à travers les *Brassicaceae*, telles que établies dans Haudry et al. [57]. Les CNS sont majoritairement de petite taille (aux alentours de 20 paires de bases) et pourraient en principe correspondre à des cibles potentielles de miARN. Nous avons donc déterminé la proportion des cibles potentielles de miARN dans les régions intergéniques qui chevauchent un CNS. C'est l'objet du Tableau 3.5. Nous observons que 7,0% et 6,4% des cibles potentielles intergéniques chevauchent un CNS pour *A. thaliana* et *A. lyrata* respectivement. Ces valeurs passent à 12,8% et 13,0% lorsque l'on s'intéresse uniquement aux miARN fiables. Ces valeurs sont significativement supérieures à l'attendu aléatoire ($p < 0.001$), sauf pour le mélange dinucléotide local chez *A. thaliana* avec tous les miARN ($p = 0.318$) (Figure 3.6 et Tableau 3.5). Il semblerait donc qu'une partie des cibles potentielles de miARN soit des régions conservées à longue échelle de temps, ce qui suggère qu'elles pourraient posséder un rôle fonctionnel important.

Dans un second temps, nous avons cherché à savoir quel pouvait être la généralité d'un nouveau mode d'action des miARN récemment identifié au locus

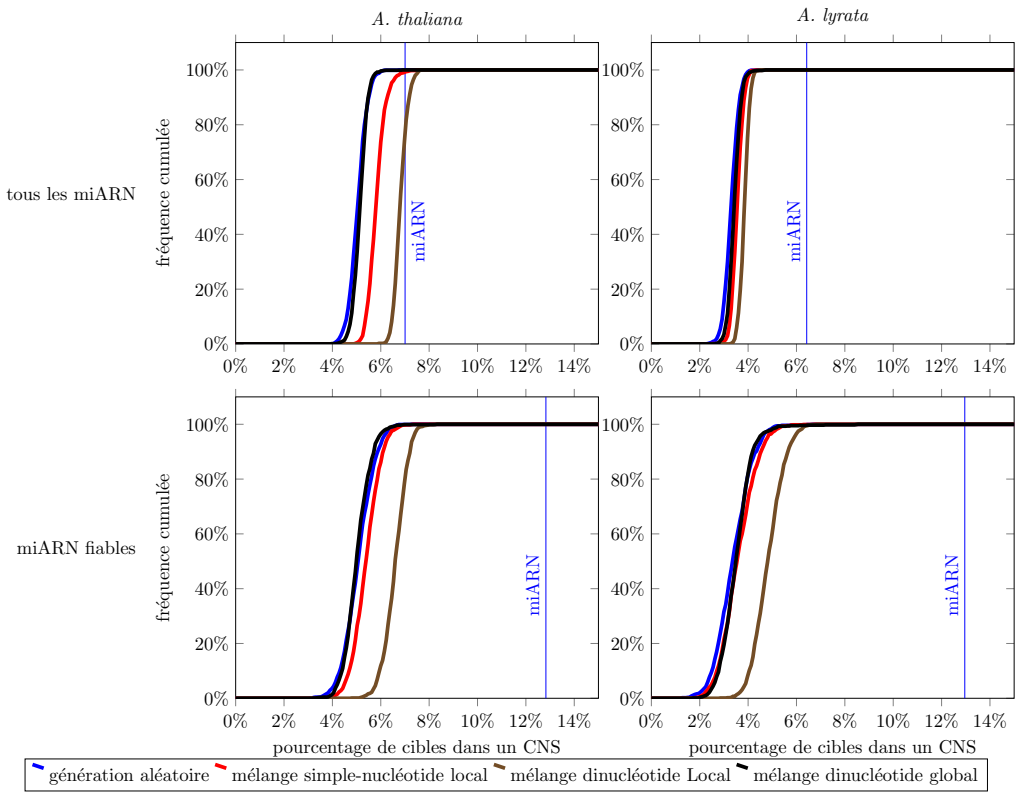



FIGURE 3.6 – Comparaison du pourcentage moyen des cibles potentielles et attendues localisées dans un CNS chez*A. thaliana* (gauche) et *A. lyrata* (à droite) pour l'ensemble des miARN (en haut) et l'ensemble restreint des miARN «fiables». La valeur observée est représentée par une barre verticale sur chaque graphe et la fréquence cumulée attendue sous différentes hypothèses nulles (voir sous-section 3.1.2) est représentée par des courbes de couleur.

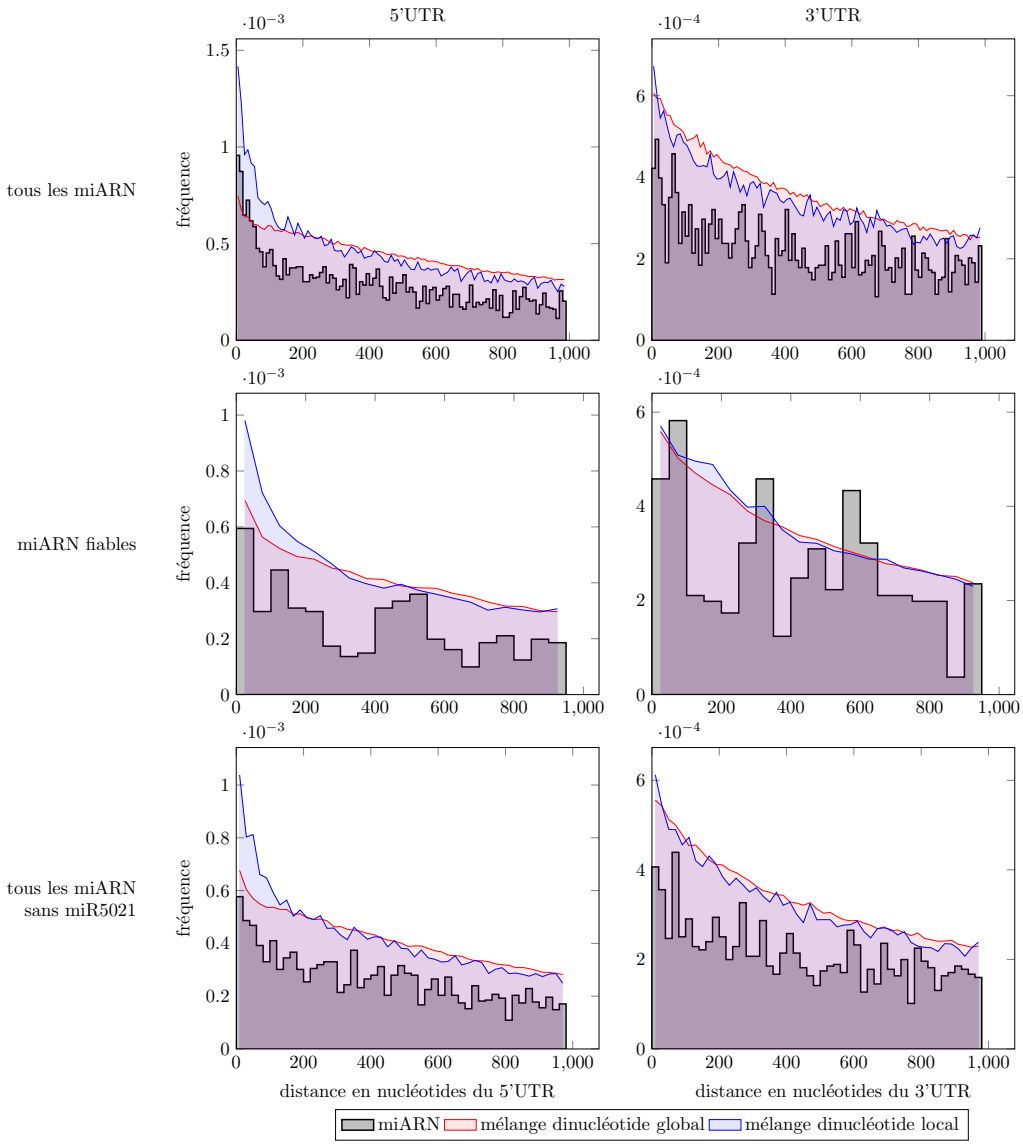



FIGURE 3.7 – Comparaison de la distribution des distances à l'UTR 5' (à gauche) et l'UTR 3' (à droites) les plus proches des cibles potentielles et attendues à 95% dans le génome d'*A. thaliana* pour l'ensemble des miARN (en haut), l'ensemble des miARN sans mi5021 (en bas) et l'ensemble restreint des miARN «fiables». La fréquence des valeurs observées est représenté par les histogrammes gris clairs sur chaque graphe et la fréquence attendue sous différentes hypothèses nulles (voir sous-section 3.1.2) est représentée par des courbes de couleur. Dans le cadre des hypothèses nulles, la courbes représente 95% des valeurs les plus basses dans chaque classe.

| | miARN | % de cibles dans un CNS | p-valeur | | | |
|---|---|---|---|---|---|---|
| | | | GA | SNL | DNL | DNG |
| *A. thaliana* | tous | 7,00% | <0,001 | 0,011 | 0,318 | <0,001 |
| | fiables | 12,83% | <0,001 | <0,001 | <0,001 | <0,001 |
| *A. lyrata* | tous | 6,42% | <0,001 | <0,001 | <0,001 | <0,001 |
| | fiables | 12,96% | <0,001 | <0,001 | <0,001 | <0,001 |

Tableau 3.5 – Comparaison du pourcentage des cibles localisées dans un CNS chez *A. thaliana* et *A. lyrata* pour l'ensemble des miARN (tous) et l'ensemble restreint des miARN «fiables» (fiables). Les p-valeurs sont obtenues par différentes procédures de génération aléatoire des séquences selon l'hypothèse nulle «la proportion de cibles dans un CNS n'est pas supérieur à l'attendu aléatoire». GA : génération aléatoire, SNL : mélange simple-nucléotide local, DNL : mélange dinucléotide local , DNG : mélange dinucléotide global.

d'auto-incompatibilité chez *Arabidopsis*. Selon ce modèle, les miARN identifiés ont une cible en amont immédiat du gène qu'ils régulent [159]. Au vu de l'importance des cibles potentielles dans les régions intergéniques conservées, il se pourrait que ce mode d'action soit en fait plus courant que ce qui est actuellement connu. De façon à déterminer si les cibles potentielles dans les régions intergéniques correspondent à ce type de ciblage, nous avons analysé la distance des cibles intergéniques à leurs gènes voisins et recherché un enrichissement en cibles en amont des gènes. Pour chaque cible prédite dans une région intergénique, nous avons recherché le gène qui présente son extrémité 5' à cette cible le plus proche et calculé la distance le séparant de celle-ci (en nombre de nucléotides) Nous avons fait de même pour l'extrémité 3'. Ainsi pour chaque cible, nous avons donc identifié 2 gènes qu'elle pourrait potentiellement réguler, *via* des effets en 5' ou en 3'.

Chez *A. thaliana*, nous observons que les cibles potentielles de miARN sont significativement plus présentes tout près de la région 5'UTR par rapport au mélange dinucléotide global, avec un clair enrichissement des cibles potentielles dans les 80 paires de bases en amont des gènes (Figure 3.7). A première vue, cela semblerait indiquer que le mode de ciblage original observé au locus S pourrait être largement répandu et non spécifiquement restreint au cas documenté par Tarutani et al. [159]. Cependant plusieurs éléments permettent de douter de

cette possibilité. D'une part, le mélange de dinucléotide local ne montre pas cette écart ce qui indique qu'il ne s'agit pas d'une propriété globale de l'ensemble des miARN mais d'une composition en dinucléotides spécifique de certains miARN. En accord avec cette hypothèse, le signal d'enrichissement disparaît lorsque l'on restreint l'analyse aux miARN dits «fiables». De façon plus précise encore, le retrait de miR5021 à lui seul cause la perte de ce signal. Il semble donc que cet enrichissement soit spécifiquement dû à ce seul miARN à la composition nucléotidique particulière (voir section 3.4).

Chez *A. lyrata*, le signal semble être absent lorsque l'on s'intéresse à l'ensemble des miARN. Au contraire, nous observons un enrichissement en cibles par rapport à l'attendu en amont des gènes lorsque seuls les miARN fiables sont considérés.

De façon intéressante, la région 3'UTR ne montre aucun signe d'enrichissement en aval immédiat des gènes chez les deux espèces. Mais, pour chacune des deux espèces, une accumulation de cibles aux alentours de 600 nucléotides en aval des gènes est observées, dont la pertinence reste à déterminer.

## 3.4 Discussion

L'utilisation de Bwolo pour la recherche de cibles potentielles de miARN nous a permis de réaliser des études statistiques sur le génome complet. C'est grâce à son efficacité dans la recherche de régions similaires qu'il nous a été possible de réaliser un grand nombre de réplicats avec des séquences aléatoires.

Notre analyse a révélé l'existence de nombreuses cibles potentielles de miARN dans le génome des deux espèces. Bien que les règles précises distinguant une cible potentielle d'une cible réellement fonctionnelle ne soient à ce jour pas bien comprises, et qu'il soit donc difficile d'établir une correspondance directe entre la présence d'une séquence fortement identique et celle d'une cible potentielle, cette observation illustre le potentiel des miARN à réguler un très grand nombre de sites cibles. On observe plus de cibles potentielles de miARN qu'attendu au hasard dans les deux espèces. Deux explications permettent de rendre compte de cette observation. D'une part, il est possible que la sélection ait favorisé au cours de l'évolution le recrutement de cibles dans tout le génome, ce qui implique

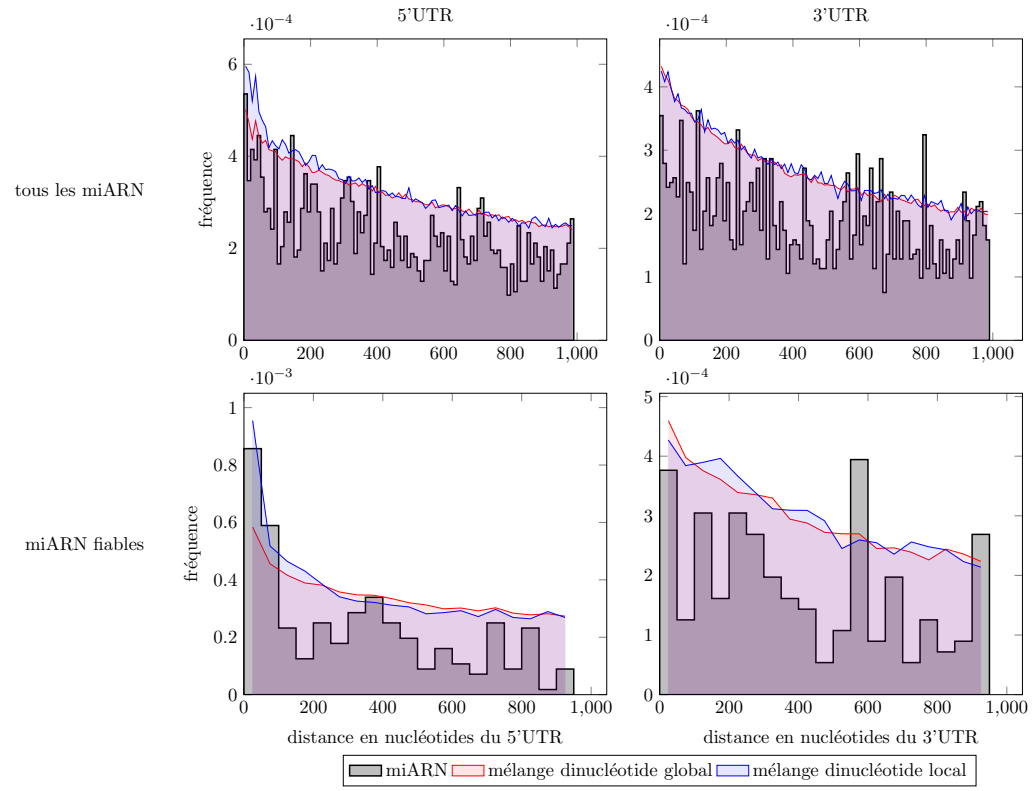



FIGURE 3.8 – Comparaison de la distribution des distances à l'UTR 5' (à gauche) et l'UTR 3' (à droites) les plus proches des cibles potentielles et attendues à 95% dans le génome d'*A. lyrata* pour l'ensemble des miARN (en haut), et l'ensemble restreint des miARN «fiables» (en bas). La fréquence des valeurs observées est représenté par les histogrammes gris clairs sur chaque graphe et la fréquence attendue sous différentes hypothèses nulles (voir sous-section 3.1.2) est représentée par des courbes de couleur. Dans le cadre des hypothèses nulles, la courbes représente 95% des valeurs les plus basses dans chaque classe.

une sélection forte et un rôle fonctionnel au moins pour une partie d'entre elles. D'autre part, il est également possible que cet enrichissement en cibles soit dû à l'origine même des miARN, plusieurs indices suggérant que les miARN pourraient être initialement formés à partir d'éléments transposables eux même fortement répétés [126, 136].

De façon surprenante, malgré une taille de génome 40% inférieure chez *A. thaliana* (125Mb) par rapport à *A. lyrata* (207Mb), la distribution du nombre de cibles par miARN est très similaire, voire légèrement décalée vers des valeurs supérieures chez *A. thaliana*. *A. thaliana* subit une réduction toujours en cours de la taille de son génome, due à une sélection importante en faveur de très nombreuses délétions de petite taille majoritairement dans les régions non codantes et les transposons [61]. Les régions géniques étant enrichies en cibles, mais dans des proportions moindres qu'attendues au hasard, nos résultats suggèrent que ces régions perdues chez *A. thaliana* comportaient relativement peu de cibles de miARN, celles-ci ayant été spécifiquement préservées des délétions. Cette conservation, de façon intéressante, s'étend à plus grande échelle à de nombreuses cibles potentielles ayant également été préservées à travers de multiples *Brassicaceae* [57]. Là encore, le processus permettant cette conservation reste à déterminer, mais pourrait impliquer un rôle fonctionnel de ces cibles intergéniques malgré le fait qu'elles ne correspondent pas au mode de ciblage canonique des miARN.

Un exemple récent de cible de miARN non exoniques a été expérimentalement validé dans l'équipe [36] ce qui pose la question de la généralité de ce mode de ciblage. Nos résultats montrent un enrichissement global en cibles potentielles dans les 80 paires de bases en amont des gènes, mais cet enrichissement n'est visible chez *A. thaliana* que sur l'ensemble global des miARN, mais pas sous le sous-ensemble des miARN fiables tandis que l'inverse est observé chez *A. lyrata*. A ce stade, nos résultats suggèrent donc que le mode de ciblage en amont des gènes n'est probablement pas une propriété générale des miARN. Une seule exception semble être détectée. En effet, l'enrichissement détecté chez *A. thaliana* est intégralement dû à un seul miARN, miR5021b qui possède de nombreuses cibles dans les promoteurs des gènes et dont la composition nucléotidique est très particulière. Sa séquence répétée correspond à un site de fixation

(élément cis) appelé *TL1* (GAAGAAGAA) d'un facteur de transcription TBF1. Le profil d'expression de TBF1 et la nature des gènes qu'il régule suggèrent qu'il pourrait jouer un rôle dans mise en œuvre de la défense contre les pathogènes et la réponse au stress thermique [120, 171]. Il serait maintenant intéressant de déterminer si miR5021 est effectivement capable de moduler l'expression des gènes comportant cet élément cis. Une stratégie possible pourrait consister à caractériser des plantes mutantes dont miR5021 est inactivé en ce qui concerne leur phénotype de résistance aux pathogènes et/ou l'expression des gènes possédant *TL1*. Les miARN identifiés par Durand et al. semblent réprimer l'expression de leur gène cible en provoquant sa méthylation [159], ce qui suggère une autre approche expérimentale consistant à examiner le profil de méthylation des gènes possédant une cible potentielle de miR5021.

L'analyse des données présentées ici ouvre de nombreuses perspectives. D'une part nous nous sommes intéressés à seulement deux espèces du genre *Arabidopsis*. Les outils étant maintenant disponibles, il serait intéressant d'étendre cette étude à l'ensemble croissant des espèces de plantes pour lesquelles un génome de référence annoté est disponible et pour lesquelles le répertoire de miARN est connu.

Les CNS utilisées ici sont des sites conservés à l'échelle des *Brassicaceae*. A cette échelle phylogénétique, il serait maintenant intéressant de comparer les vitesses d'évolution des régions cibles des régions non cibles pour les cibles canoniques. De la même façon, il serait aussi intéressant de comparer les statistiques de polymorphisme et de divergence au sein d'une même espèce entre les régions cibles et les régions non cibles. De nombreuses études ont en effet récemment mis en évidence que les positions nucléotidiques synonymes au sein des gènes (dont la variation n'affecte pas l'acide aminé codé) pouvaient subir des contraintes fonctionnelles dont la nature est souvent mal connue mais pouvait inclure la présence de signaux d'épissage [140] , le positionnement des nucléosomes [172], la disponibilité en ARN de transfert [64], la présence de site de fixation de facteur de transcription [154, 181], ou l'absence de structures secondaires diminuant l'accessibilité des cibles potentielles de miARN [54]. Le

rôle relatif de ces facteurs est mal connu [22] mais l'effet général peut être fort [91]. Il serait intéressant de déterminer si la présence des cibles potentielles que j'ai identifiées peut également constituer une contrainte fonctionnelle détectable.

Enfin, durant cette étude, nous nous sommes intéressés à l'ensemble des miARN et à l'ensemble restreint des miARN fiables. Il serait intéressant de déterminer si les miARN les plus récents et les plus anciens [40, 102] partagent des propriétés uniques en termes de nombre de cibles ou de distribution à travers le génome.

# Conclusion Générale

Les miARN sont des éléments génétiques découverts récemment et leur étude constitue un domaine de recherche particulièrement riche et actif. À partir d'un modèle de base relativement simple, la connaissance du monde des petits ARN s'est affinée, révélant à la fois des interactions inattendues entre voies métaboliques initialement décrites comme distinctes et l'existence de nouveaux modes d'action ne correspondant pas au mode d'action canonique de clivage des ARNm. En particulier, la découverte récente de cibles de miARN en dehors des gènes demande de prédire les cibles à l'échelle d'un génome entier et non à l'échelle de la seule fraction exonique. Cette tâche représente un défi algorithmique et informatique pour permettre la recherche de petites séquences d'une vingtaine de nucléotides dans un génome complet et avec un taux d'erreur assez élevé.

## Amélioration des graines 01*0

Dans ce contexte, j'ai développé un nouvel algorithme de recherche de motifs approchée à partir de nouvelles graines, les graine 01*0. Ces graines ont comme caractéristique de contenir l'explosion combinatoire due au nombre d'erreurs en maîtrisant la disposition de celles-ci. Elles ont été mises en œuvre dans le logiciel Bwolo, qui permet de faire une recherche de motifs dans un grand texte tel un génome de manière particulièrement efficace. Ce travail a été publié dans[168, 169].

Nous pensons que les graines 01*0 possèdent de vraies bonnes propriétés, qui devraient en faire un concept prometteur pour d'autres domaines d'application

et permettre de nombreuses évolutions. Elles peuvent être utilisées pour la recherche d'une collection de séquences, compilées dans une banque de données, dans un texte de quelques centaines ou milliers de nucléotides, comme un gène. Il est alors possible d'indexer les motifs de la base de données en suivant la structure des graines 01*0. C'est ce que propose Piccolo, un logiciel développé par un stagiaire de master (Sébastien Bini) dans le prolongement de mon travail de thèse. Piccolo indexe les parties exactes des graines dans une table et commence par rechercher les parties exactes dans la requête puis vérifie si les parties inexactes centrales de la graines 01*0 sont valides pour enfin finir l'extension [169]. Ce programme sert par exemple à localiser de potentiels miARN conservés, issus de miRBase, dans une transcrit primaire. Il est aussi possible d'envisager les graines 01*0 sans indexer ni les motifs ni le texte, avec un algorithme à base d'automate ou de bit-parallélisme, en tirant profit de l'architecture des ordinateurs qui travaille avec des mots machines pour paralléliser plusieurs opérations d'un algorithme de recherche de motif dans une seule opération pour le processeur. Ce type d'approche pourrait permettre d'atteindre d'excellentes performances sans même avoir besoin d'utiliser un index, surtout dans les cas où son utilisation n'est pas pertinente.

L'architecture même des graines 01*0 pourrait être sujette à des extensions. En effet, à l'heure actuelle, le motif est découpé en parties de mêmes longueurs à un caractère près. Cependant, il est facile de remarquer que chaque partie n'est pas représentée dans le même nombre de sous-graines. Les parties au centre du motif sont utilisées par davantage de sous-graines que les parties se trouvant aux extrémités, car plus souvent recherchées avec une erreur. La taille de chaque partie peut alors avoir une influence sur le pouvoir de filtrage de la graine. Dans la même optique d'améliorer le pouvoir de filtrage, il peut être envisageable de tirer profit de la différence de fréquences de certains mots dans le texte pour adapter le découpage des parties. Il serait alors possible d'imaginer par exemple d'optimiser la taille des parties en fonction de leur position dans le motif, à l'image de ce qui a été fait par Russo et al. [137], mais aussi par rapport à leur fréquence dans le texte. De plus, nous n'avons finalement considéré que le motif 01*0 pour réaliser les graines. Une autre question que nous nous posons est de savoir si ce motif n'est pas un cas particulier d'un motif plus général comme

0(1*0)*. Il serait alors intéressant d'approfondir le modèle afin de vérifier si d'autres motifs de graines sont possibles et s'ils sont plus filtrants.

Une question qui reste ouverte est le pouvoir filtrant des graines 01*0. Par exemple, il a été établi que dans le cadre d'une distance de Hamming (substitution uniquement) le pouvoir filtrant des graines 01*0 est supérieur aux graines espacées [169]. Cependant, le calcul théorique du pouvoir filtrant pour une distance de Levenshtein (substitutions, insertions et délétions) se révèle plus ardu. Caractériser et comparer l'efficacité théorique des graines 01*0 par rapport à d'autres types de graines est une piste de recherche qui reste à explorer. Cependant, nous somme déjà capables de donner une limite haute du pouvoir filtrant des graines 01*0.

## Vers un outil adapté à un contexte biologique plus large

Les graines 01*0 reposent sur un modèle particulièrement simple pour l'interaction du miARN avec sa cible. Il est donc tendancieux de considérer Bwolo comme un outil spécialisé dans la prédiction de cibles canoniques, comme le sont Tapir ou TargetFinder. Dans cette perspective, deux pistes d'amélioration sont possibles. La première consiste à utiliser Bwolo comme un outil de pré-filtrage, combiné à un outil implémentant une approche *ad hoc*, telles que celles décrites dans le Chapitre 1. Une autre possibilité est de ne pas considérer l'intégralité des sous-graines et d'éliminer celles présentant, par exemple, des erreurs au niveau de la région de forte complémentarité du miARN et de sa cible, en 5'. De plus, la prise en compte des interactions G-U pourrait être réalisée en s'appuyant sur l'idée proposée dans GUUGle [46], qui utilise un FM-Index comme Bwolo, afin de traiter ces cas sans augmentation significative du temps d'exécution.

Dans un tout autre contexte, alors que les séquenceurs de seconde génération produisaient de nombreuses petites lectures de séquençage avec un taux faible d'erreur, les séquenceurs de troisième génération comme le minION ou PacBio RS II, bien que capables de produire des lectures significativement plus grandes ont un taux d'erreur de 10 à 15% dont pour la plupart des insertions et des

délétions [133]. Ce fort taux d'erreur rend la plupart (si ce n'est la totalité) des outils dédiés aux données actuelles inadaptés aux données du futur. Or les graines 01*0 ont montré leur efficacité pour un tel taux d'erreur avec la prise en compte des insertions et des délétions. Elles semblent donc toutes indiquées pour la conception d'outils travaillant sur les données de séquençage de troisième génération.

# Bibliographie


[1] M. I. Abouelhoda, S. Kurtz et E. Ohlebusch. « Replacing suffix trees with enhanced suffix arrays ». In : *Journal of Discrete Algorithms* 2.1 (2004), p. 53–86.

[2] C. Addo-Quaye, W. Miller et M. J. Axtell. « CleaveLand : a pipeline for using degradome data to find cleaved small RNA targets ». In : *Bioinformatics* 25.1 (2009), p. 130–131.

[3] C. Addo-Quaye et al. « Endogenous siRNA and microRNA targets identified by sequencing of the Arabidopsis degradome ». In : *Current biology : CB* 18.10 (2008), p. 758–762.

[4] E. Allen et al. « microRNA-Directed Phasing during Trans-Acting siRNA Biogenesis in Plants ». In : *Cell* 121.2 (2005), p. 207–221.

[5] S. F. Altschul et B. W. Erickson. « Significance of nucleotide sequence alignments : a method for random sequence permutation that preserves dinucleotide and codon usage. » In : *Molecular Biology and Evolution* 2.6 (1985), p. 526–538.

[6] S. F. Altschul et al. « Basic local alignment search tool ». In : *Journal of Molecular Biology* 215.3 (1990), p. 403–410.

[7] Arabidopsis Genome Initiative. « Analysis of the genome sequence of the flowering plant Arabidopsis thaliana ». In : *Nature* 408.6814 (2000), p. 796–815.

[8] S. A. Atambayeva, V. A. Khailenko et A. T. Ivashchenko. « Intron and exon length variation in Arabidopsis, rice, nematode, and human ». In : *Molecular Biology* 42.2 (2008), p. 312–320.

[9] M. J. Axtell et J. L. Bowman. « Evolution of plant microRNAs and their targets ». In : *Trends in Plant Science* 13.7 (2008), p. 343–349.

[10] M. J. Axtell, J. O. Westholm et E. C. Lai. « Vive la différence : biogenesis and evolution of microRNAs in plants and animals ». In : *Genome Biology* 12.4 (2011), p. 221.

[11] T. Babak, B. J. Blencowe et T. R. Hughes. « Considerations in the identification of functional RNA structural elements in genomic alignments ». In : *BMC Bioinformatics* 8 (2007), p. 33.

[12] R. Baeza-Yates et G. Gonnet. « A New Approach to Text Searching. » In : *Communications of the ACM* 35.10 (1992), p. 74–82.

[13] R. A. Baeza-Yates et C. H. Perleberg. « Fast and practical approximate string matching ». In : *Information Processing Letters* 59.1 (1996), p. 21–27.

[14] D. Belazzougui et al. « Versatile Succinct Representations of the Bidirectional Burrows-Wheeler Transform ». In : *Algorithms – ESA 2013*. Sous la dir. de H. L. Bodlaender et G. F. Italiano. Lecture Notes in Computer Science 8125. 2013, p. 133–144.

[15] E. Bernstein et al. « Role for a bidentate ribonuclease in the initiation step of RNA interference ». In : *Nature* 409.6818 (2001), p. 363–366.

[16] K. Bobiwash, S. T. Schultz et D. J. Schoen. « Somatic deleterious mutation rate in a woody plant : estimation from phenotypic data ». In : *Heredity* 111.4 (2013), p. 338–344.

[17] N. G. Bologna et al. « A loop-to-base processing mechanism underlies the biogenesis of plant microRNAs miR319 and miR159 ». In : *The EMBO Journal* 28.23 (2009), p. 3646–3656.

[18] E. Bonnet et al. « TAPIR, a web server for the prediction of plant microRNA targets, including target mimics ». en. In : *Bioinformatics* 26.12 (2010), p. 1566–1568.

[19] P. Brodersen et al. « Widespread Translational Inhibition by Plant miRNAs and siRNAs ». en. In : *Science* 320.5880 (2008), p. 1185–1190.

[20] C. Brousse et al. « A non-canonical plant microRNA target site ». en. In : *Nucleic Acids Research* 42.8 (2014), p. 5270–5279.

[21] M. Burrows et D. J. Wheeler. *A block-sorting lossless data compression algorithm.* Rapp. tech. 124. Digital Equipment Corporation, Palo Alto, California, 1994.

[22] J. V. Chamary, J. L. Parmley et L. D. Hurst. « Hearing silence : non-neutral evolution at synonymous sites in mammals ». In : *Nature Reviews Genetics* 7.2 (2006), p. 98–108.

[23] N. Chomsky. « On certain formal properties of grammars ». In : *Information and Control* 2.2 (1959), p. 137–167.

[24] R. Collobert et S. Bengio. « SVMTorch : Support Vector Machines for Large-Scale Regression Problems ». In : *Journal of Machine Learning Research* 1.2 (2001), p. 143–160.

[25] F. Crick. « Central Dogma of Molecular Biology ». In : *Nature* 227.5258 (1970), p. 561–563.

[26] M. Crochemore et al. « Computing the Burrows–Wheeler transform in place and in small space ». In : *Journal of Discrete Algorithms*. StringMasters 2012 & 2013 Special Issue (Volume 2) 32 (2015), p. 44–52.

[27] J. T. Cuperus, N. Fahlgren et J. C. Carrington. « Evolution and Functional Diversification of MIRNA Genes ». In : *The Plant Cell* 23.2 (2011), p. 431–442.

[28] X. Dai et P. X. Zhao. « psRNATarget : a plant small RNA target analysis server ». In : *Nucleic Acids Research* 39 (suppl 2 2011), W155–W159.

[29] X. Dai, Z. Zhuang et P. X. Zhao. « Computational analysis of miRNA targets in plants : current status and challenges ». en. In : *Briefings in Bioinformatics* 12.2 (2011), p. 115–121.

[30] R. B. Darnell. « HITS-CLIP : panoramic views of protein-RNA regulation in living cells ». In : *Wiley interdisciplinary reviews. RNA* 1.2 (2010), p. 266–286.

[31] J. Ding et al. « Genome-wide search for miRNA-target interactions in Arabidopsis thaliana with an integrated approach ». In : *BMC Genomics* 13.Suppl 3 (2012), S3.

[32] J. G. Doench et P. A. Sharp. « Specificity of microRNA target selection in translational repression ». en. In : *Genes & Development* 18.5 (2004), p. 504–511.

[33] E. A. Doherty et J. A. Doudna. « Ribozyme Structures and Mechanisms ». In : *Annual Review of Biochemistry* 69.1 (2000), p. 597–615.

[34] A. Döring et al. « SeqAn An efficient, generic C++ library for sequence analysis ». In : *BMC Bioinformatics* 9.1 (2008), p. 11.

[35] M. Dsouza, N. Larsen et R. Overbeek. « Searching for patterns in genomic data ». In : *Trends in Genetics* 13.12 (1997), p. 497–498.

[36] E. Durand et al. « Dominance hierarchy arising from the evolution of a complex small RNA regulatory network ». In : *Science* 346.6214 (2014), p. 1200–1205.

[37] S. M. Elbashir. « Functional anatomy of siRNAs for mediating efficient RNAi in Drosophila melanogaster embryo lysate ». In : *The EMBO Journal* 20.23 (2001), p. 6877–6888.

[38] A. J. Enright et al. « MicroRNA targets in Drosophila ». en. In : *Genome Biology* 5.1 (2003), R1.

[39] M. Esteller. « Non-coding RNAs in human disease ». In : *Nature Reviews Genetics* 12.12 (2011), p. 861–874.

[40] N. Fahlgren et J. C. Carrington. « miRNA Target Prediction in Plants ». In : *Methods in Molecular Biology (Clifton, N.J.)* 592 (2010), p. 51–57.

[41] N. Fahlgren et al. « High-Throughput Sequencing of Arabidopsis microRNAs : Evidence for Frequent Birth and Death of MIRNA Genes ». In : *PLoS ONE* 2.2 (2007), e219.

[42] P. Ferragina et G. Manzini. « Indexing Compressed Text ». In : *J. ACM* 52.4 (2005), p. 552–581.

[43] P. Ferragina et al. « Compressed representations of sequences and full-text indexes ». In : *ACM Transactions on Algorithms* 3 (2007).

[44] P. Ferragina et al. « Compressed text indexes : From theory to practice ». In : *Journal of Experimental Algorithmics (JEA)* 13 (2009), p. 12.

[45] Z. Galil et K. Park. « An Improved Algorithm For Approximate String Matching ». In : *SIAM Journal on Computing* 19.6 (1990), p. 989–999.

[46] W. Gerlach et R. Giegerich. « GUUGle : a utility for fast exact matching under RNA complementary rules including G–U base pairing ». In : *Bioinformatics* 22.6 (2006), p. 762–764.

[47] M. A. German et al. « Global identification of microRNA–target RNA pairs by parallel analysis of RNA ends ». en. In : *Nature Biotechnology* 26.8 (2008), p. 941–946.

[48] O. Gotoh. « An improved algorithm for matching biological sequences ». In : *Journal of Molecular Biology* 162.3 (1982), p. 705–708.

[49] S. Griffiths-Jones et al. « miRBase : tools for microRNA genomics ». In : *Nucleic Acids Research* 36 (suppl 1 2008), p. D154–D158.

[50] S. Griffiths-Jones et al. « Rfam : annotating non-coding RNAs in complete genomes ». In : *Nucleic Acids Research* 33 (suppl 1 2005), p. D121–D124.

[51] A. Grishok et al. « Genes and Mechanisms Related to RNA Interference Regulate Expression of the Small Temporal RNAs that Control C. elegans Developmental Timing ». In : *Cell* 106.1 (2001), p. 23–34.

[52] R. Grossi, A. Gupta et J. S. Vitter. « High-order entropy-compressed text indexes ». In : *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. 2003, p. 841–850.

[53] R. Grossi et J. S. Vitter. « Compressed suffix arrays and suffix trees with applications to text indexing and string matching ». In : *SIAM Journal on Computing* 35.2 (2005), p. 378–407.

[54] W. Gu et al. « Selection on Synonymous Sites for Increased Accessibility around miRNA Binding Sites in Plants ». In : *Molecular Biology and Evolution* 29.10 (2012), p. 3037–3044.

[55] C. Guerrier-Takada et al. « The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme ». In : *Cell* 35.3, Part 2 (1983), p. 849–857.

[56] B. Haley et P. D. Zamore. « Kinetic analysis of the RNAi enzyme complex ». en. In : *Nature Structural & Molecular Biology* 11.7 (2004), p. 599–606.

[57] A. Haudry et al. « An atlas of over 90,000 conserved noncoding sequences provides insight into crucifer regulatory regions ». In : *Nature Genetics* 45.8 (2013), p. 891–898.

[58] J. Hausser et al. « Relative contribution of sequence and structure features to the mRNA binding of Argonaute/EIF2C–miRNA complexes and the degradation of miRNA targets ». In : *Genome Research* 19.11 (2009), p. 2009–2020.

[59] R. Heikham et R. Shankar. « Flanking region sequence information to refine microRNA target predictions ». In : *Journal of Biosciences* 35.1 (2010), p. 105–118.

[60] I. L. Hofacker et al. « Fast folding and comparison of RNA secondary structures ». en. In : *Monatshefte für Chemie / Chemical Monthly* 125.2 (1994), p. 167–188.

[61] T. T. Hu et al. « The Arabidopsis lyrata genome sequence and the basis of rapid genome size change ». In : *Nature genetics* 43.5 (2011), p. 476–481.

[62] G. Hutvágner et al. « A Cellular Function for the RNA-Interference Enzyme Dicer in the Maturation of the let-7 Small Temporal RNA ». In : *Science* 293.5531 (2001), p. 834–838.

[63] H. Hyyrö. « A Bit-vector Algorithm for Computing Levenshtein and Damerau Edit Distances ». In : *Nordic J. of Computing* 10.1 (2003), 29–39.

[64] K. Iida et H. Akashi. « A test of translational selection at 'sites in the human genome : base composition comparisons in alternatively spliced genes ». In : *Gene*. Papers presented at the Anton Dohrn Workshop 261.1 (2000), p. 93–105.

[65] A. Itaya et al. « A Structured Viroid RNA Serves as a Substrate for Dicer-Like Cleavage To Produce Biologically Active Small RNAs but Is Resistant to RNA-Induced Silencing Complex-Mediated Degradation ». In : *Journal of Virology* 81.6 (2007), p. 2980–2994.

[66] A. Jha et R. Shankar. « Employing machine learning for reliable miRNA target identification in plants ». In : *BMC Genomics* 12 (2011), p. 636.

[67] B. John et al. « Human MicroRNA Targets ». In : *PLoS Biol* 2.11 (2004), e363.

[68] M. W. Jones-Rhoades et D. P. Bartel. « Computational Identification of Plant MicroRNAs and Their Targets, Including a Stress-Induced miRNA ». In : *Molecular Cell* 14.6 (2004), p. 787–799.

[69] M. W. Jones-Rhoades, D. P. Bartel et B. Bartel. « MicroRNAs AND THEIR REGULATORY ROLES IN PLANTS ». In : *Annual Review of Plant Biology* 57.1 (2006), p. 19–53.

[70] F. V. Karginov et al. « A biochemical approach to identifying microRNA targets ». en. In : *Proceedings of the National Academy of Sciences* 104.49 (2007), p. 19291–19296.

[71] J. Kärkkäinen et P. Sanders. « Simple linear work suffix array construction ». In : *Proc. 30th Internat. Colloq. Automata, Languages & Programming*. 2003, p. 943–955.

[72] J. Kärkkäinen. « Fast BWT in small space by blockwise suffix sorting ». In : *Theoretical Computer Science*. The Burrows-Wheeler Transform 387.3 (2007), p. 249–257.

[73] J. Kärkkäinen et J. C. Na. « Faster Filters for Approximate String Matching ». In : *ALENEX*. SIAM. 2007, p. 84–90.

[74] U. Keich et al. « On spaced seeds for similarity search ». In : *Discrete Applied Mathematics* 138 (3) (2004), p. 253–263.

[75] B. Khraiwesh, J.-K. Zhu et J. Zhu. « Role of miRNAs and siRNAs in biotic and abiotic stress responses of plants ». In : *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms*. Plant gene regulation in response to abiotic stress 1819.2 (2012), p. 137–148.

[76] D. Kim et al. « Linear-time construction of suffix arrays ». In : *Proc. 14th Annual Symp. Combinatorial pattern matching*. 2003, p. 186–199.

[77] P. Ko et S. Aluru. « Space efficient linear time construction of suffix arrays ». In : *Proc. 14th Annual Symp. Combinatorial pattern matching*. 2003, p. 200–210.

[78] A. Kozomara et S. Griffiths-Jones. « miRBase : annotating high confidence microRNAs using deep sequencing data ». In : *Nucleic Acids Research* 42 (D1 2014), p. D68–D73.

[79] J. Krüger et M. Rehmsmeier. « RNAhybrid : microRNA target prediction easy, fast and flexible ». en. In : *Nucleic Acids Research* 34.suppl 2 (2006), W451–W454.

[80] K. Kruger et al. « Self-splicing RNA : autoexcision and autocyclization of the ribosomal RNA intervening sequence of Tetrahymena ». eng. In : *Cell* 31.1 (1982), p. 147–157.

[81] G. Kucherov, L. Noé et M. Roytberg. « Subset seed automaton ». In : *International Conference on Implementation and Application of Automata (CIAA)*. Lecture Notes in Computer Science 4783. 2007, p. 180–191.

[82] G. Kucherov, L. Noé et M. A. Roytberg. « A Unifying Framework for Seed Sensitivity and its Application to Subset Seeds ». In : *J. Bioinformatics and Computational Biology* 4.2 (2006), p. 553–570.

[83] D. E. Kuhn et al. « Experimental Validation of miRNA Targets ». In : *Methods (San Diego, Calif.)* 44.1 (2008), p. 47–54.

[84] S. Kurtz. « Reducing the space requirement of suffix trees ». In : *Software—Practice and Experience* 29.13 (1999), p. 1149–1171.

[85] M. Lagos-Quintana et al. « Identification of Novel Genes Coding for Small Expressed RNAs ». In : *Science* 294.5543 (2001), p. 853–858.

[86] E. C. Lai. « Micro RNAs are complementary to 3' UTR sequence motifs that mediate negative post-transcriptional regulation ». In : *Nature Genetics* 30.4 (2002), p. 363–364.

[87] P. Lamesch et al. « The Arabidopsis Information Resource (TAIR) : improved gene annotation and new tools ». In : *Nucleic Acids Research* 40 (D1 2012), p. D1202–D1210.

[88] G. M. Landau et U. Vishkin. « Fast string matching with k differences ». In : *Journal of Computer and System Sciences* 37.1 (1988), p. 63–78.

[89] B. Langmead et S. L. Salzberg. « Fast gapped-read alignment with Bowtie 2 ». In : *Nature methods* 9.4 (2012), p. 357–359.

[90] N. C. Lau et al. « An Abundant Class of Tiny RNAs with Probable Regulatory Roles in Caenorhabditis elegans ». In : *Science* 294.5543 (2001), p. 858–862.

[91] D. S. Lawrie et al. « Strong Purifying Selection at Synonymous Sites in D. melanogaster ». In : *PLOS Genet* 9.5 (2013), e1003527.

[92] J. T. Lee. « Lessons from X-chromosome inactivation : long ncRNA as guides and tethers to the epigenome ». In : *Genes & Development* 23.16 (2009), p. 1831–1842.

[93] R. C. Lee, R. L. Feinbaum et V. Ambros. « The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14 ». In : *Cell* 75.5 (1993), p. 843–854.

[94] B. P. Lewis et al. « Prediction of Mammalian MicroRNA Targets ». In : *Cell* 115.7 (2003), p. 787–798.

[95] F. Li, R. Orban et B. Baker. « SoMART : a web server for plant miRNA, tasiRNA and target gene analysis ». In : *The Plant Journal* 70.5 (2012), p. 891–901.

[96] H. Li. « Fast construction of FM-index for long sequence reads ». eng. In : *Bioinformatics (Oxford, England)* 30.22 (2014), p. 3274–3275.

[97] H. Li et R. Durbin. « Fast and accurate short read alignment with Burrows–Wheeler transform ». In : *Bioinformatics* 25.14 (2009), p. 1754–1760.

[98] C. Llave et al. « Cleavage of Scarecrow-like mRNA Targets Directed by a Class of Arabidopsis miRNA ». en. In : *Science* 297.5589 (2002), p. 2053–2056.

[99] C. Llave et al. « Endogenous and Silencing-Associated Small RNAs in Plants ». en. In : *The Plant Cell* 14.7 (2002), p. 1605–1619.

[100] R. Lorenz et al. « ViennaRNA Package 2.0 ». en. In : *Algorithms for Molecular Biology* 6.1 (2011), p. 1–14.

[101] B. Ma, J. Tromp et M. Li. « Patternhunter —- faster and more sensitive homology search ». In : *Bioinformatics* 18 (3) (2002), p. 440–445.

[102] Z. Ma, C. Coruh et M. J. Axtell. « Arabidopsis lyrata Small RNAs : Transient MIRNA and Small Interfering RNA Loci within the Arabidopsis Genus[W][OA] ». In : *The Plant Cell* 22.4 (2010), p. 1090–1103.

[103] J. T. Madison. « Primary Structure of RNA ». In : *Annual Review of Biochemistry* 37.1 (1968), p. 131–148.

[104] V. Mäkinen. « Compact Suffix Array – A Space-Efficient Full-Text Index ». In : *Fundamenta Infomaticae, Special Issue - Computing Patterns in Strings* 56.1-2 (2003), p. 191–210.

[105] V. Mäkinen et G. Navarro. « Compressed compact suffix arrays ». In : *Proc. 15th Annual Symposium on Combinatorial Pattern Matching*. T. 3109. 2004, p. 420–433.

[106] V. Mäkinen et al. « Unified view of backward backtracking in short read mapping ». In : *Algorithms and Applications*. 2010, p. 182–195.

[107] A. C. Mallory et al. « MicroRNA control of PHABULOSA in leaf development : importance of pairing to the microRNA 5′ region ». en. In : *The EMBO Journal* 23.16 (2004), p. 3356–3364.

[108] U. Manber et G. Myers. « Suffix arrays : a new method for on-line string searches ». In : *SODA '90 : Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA, 1990, p. 319–327.

[109] J. R. McConnell et al. « Role of PHABULOSA and PHAVOLUTA in determining radial patterning in shoots ». en. In : *Nature* 411.6838 (2001), p. 709–713.

[110] E. McCreight. « A Space-Economical Suffix Tree Construction Algorithm ». In : *J. ACM* 23.2 (1976), p. 262–272.

[111] Y. Meng, C. Shao et M. Chen. « Toward microRNA-mediated gene regulatory networks in plants ». In : *Briefings in Bioinformatics* (2011), bbq091.

[112] Y. Meng et al. « Introns targeted by plant microRNAs : a possible novel mechanism of gene regulation ». In : *Rice* 6.1 (2013), p. 1–10.

[113] B. C. Meyers et al. « Criteria for Annotation of Plant MicroRNAs ». In : *The Plant Cell* 20.12 (2008), p. 3186–3190.

[114] R. Milo et al. « Network Motifs : Simple Building Blocks of Complex Networks ». In : *Science* 298.5594 (2002), p. 824–827.

[115] N. A. Moran et G. M. Bennett. « The Tiniest Tiny Genomes ». In : *Annual Review of Microbiology* 68.1 (2014), p. 195–215.

[116] G. Navarro et R. Baeza-yates. « A Hybrid Indexing Method for Approximate String Matching ». In : *Journal of Discrete Algorithms* 1 (2001), p. 2000.

[117] E. P. Nawrocki et al. « Rfam 12.0 : updates to the RNA families database ». In : *Nucleic Acids Research* 43 (D1 2015), p. D130–D137.

[118] S. B. Needleman et C. D. Wunsch. « A general method applicable to the search for similarities in the amino acid sequence of two proteins ». In : *Journal of Molecular Biology* 48.3 (1970), p. 443–453.

[119] U. A. Ørom et A. H. Lund. « Isolation of microRNA targets using biotinylated synthetic microRNAs ». In : *Methods*. MicroRNAs Part A 43.2 (2007), p. 162–165.

[120] K. M. Pajerowska-Mukhtar et al. « The HSF-like Transcription Factor TBF1 Is a Major Molecular Switch for Plant Growth-to-Defense Transition ». In : *Current Biology* 22.2 (2012), p. 103–112.

[121] J. F. Palatnik et al. « Control of leaf morphogenesis by microRNAs ». en. In : *Nature* 425.6955 (2003), p. 257–263.

[122] W. Park et al. « CARPEL FACTORY, a Dicer Homolog, and HEN1, a Novel Protein, Act in microRNA Metabolism in Arabidopsis thaliana ». In : *Current Biology* 12.17 (2002), p. 1484–1495.

[123] A. E. Pasquinelli et al. « Conservation of the sequence and temporal expression of let-7 heterochronic regulatory RNA ». In : *Nature* 408.6808 (2000), p. 86–89.

[124] W. R. Pearson et D. J. Lipman. « Improved tools for biological sequence comparison. » In : *Proceedings of the National Academy of Sciences of the United States of America* 85.8 (1988), p. 2444–2448.

[125] W. R. Pearson. « Searching protein sequence libraries : Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms ». In : *Genomics* 11.3 (1991), p. 635–650.

[126] J. Piriyapongsa et I. K. Jordan. « Dual coding of siRNAs and miRNAs by plant transposable elements ». In : *RNA* 14.5 (2008), p. 814–821.

[127] S. J. Puglisi, W. F. Smyth et A. H. Turpin. « A taxonomy of suffix array construction algorithms ». In : *acm Computing Surveys (CSUR)* 39.2 (2007), p. 4.

[128] R. Rajagopalan et al. « A diverse and evolutionarily fluid set of microRNAs in Arabidopsis thaliana ». In : *Genes & Development* 20.24 (2006), p. 3407–3425.

[129] V. Rawat et al. « Improving the Annotation of Arabidopsis lyrata Using RNA-Seq Data ». In : *PLoS ONE* 10.9 (2015), e0137391.

[130] M. Rehmsmeier et al. « Fast and effective prediction of microRNA/target duplexes ». In : *RNA* 10.10 (2004), p. 1507–1517.

[131] B. J. Reinhart et al. « MicroRNAs in plants ». In : *Genes & Development* 16.13 (2002), p. 1616–1626.

[132] B. J. Reinhart et al. « The 21-nucleotide let-7 RNA regulates developmental timing in Caenorhabditis elegans ». In : *Nature* 403.6772 (2000), p. 901–906.

[133] J. A. Reuter, D. V. Spacek et M. P. Snyder. « High-Throughput Sequencing Technologies ». In : *Molecular Cell* 58.4 (2015), p. 586–597.

[134] S. Y. Rhee et al. « The Arabidopsis Information Resource (TAIR) : a model organism database providing a centralized, curated gateway to Arabidopsis biology, research materials and community ». In : *Nucleic Acids Research* 31.1 (2003), p. 224–228.

[135] M. W. Rhoades et al. « Prediction of Plant MicroRNA Targets ». In : *Cell* 110.4 (2002), p. 513–520.

[136] J. T. Roberts, S. E. Cardin et G. M. Borchert. « Burgeoning evidence indicates that microRNAs were initially formed from transposable element sequences ». In : *Mobile Genetic Elements* 4.3 (2014), e29255.

[137] L. M. S. Russo et al. « Approximate String Matching with Compressed Indexes ». In : *Algorithms* 2.3 (2009), p. 1105–1136.

[138] K. Sadakane et H. Imai. « A cooperative distributed text database management method unifying search and compression based on the Burrows-Wheeler transformation ». In : *Lecture notes in computer science*. ER '98 workshops on data warehousing and data mining, mobile data access, and collaborative work support and spatio-temporal data management. 1999, p. 434–445.

[139] K. Sadakane. « New text indexing functionalities of the compressed suffix arrays ». In : *Journal of Algorithms* 48.2 (2003), p. 294–313.

[140] R. Savisaar et L. D. Hurst. « Purifying Selection on Exonic Splice Enhancers in Intronless Genes ». In : *Molecular Biology and Evolution* (2016), msw018.

[141] S. E. Schauer et al. « DICER-LIKE1 : blind men and elephants in Arabidopsis development ». In : *Trends in Plant Science* 7.11 (2002), p. 487–491.

[142] S. Schbath et al. « Mapping Reads on a Genomic Sequence : An Algorithmic Overview and a Practical Comparative Analysis ». en. In : *Journal of Computational Biology* 19.6 (2012), p. 796–813.

[143] T. Schnattinger, E. Ohlebusch et S. Gog. « Bidirectional Search in a String with Wavelet Trees ». In : *Combinatorial Pattern Matching*. Sous la dir. d'A. Amir et L. Parida. Lecture Notes in Computer Science 6129. 2010, p. 40–50.

[144] R. Schwab et al. « Specific Effects of MicroRNAs on the Plant Transcriptome ». In : *Developmental Cell* 8.4 (2005), p. 517–527.

[145] R. Schweet et R. Heintz. « Protein Synthesis ». In : *Annual Review of Biochemistry* 35.1 (1966), p. 723–758.

[146] P. H. Sellers. « The theory and computation of evolutionary distances : Pattern recognition ». In : *Journal of Algorithms* 1.4 (1980), p. 359–373.

[147] J. Sirén. « Burrows-Wheeler transform for terabases ». In : *arXiv :1511.00898 [cs]* (2015).

[148] G. S. C. Slater et E. Birney. « Automated generation of heuristics for biological sequence comparison ». In : *BMC Bioinformatics* 6 (2005), p. 1–11.

[149] T. F. Smith et M. S. Waterman. « Identification of common molecular subsequences ». In : *Journal of Molecular Biology* 147.1 (1981), p. 195–197.

[150] F. Spitz et E. E. M. Furlong. « Transcription factors : from enhancer binding to developmental control ». In : *Nature Reviews Genetics* 13.9 (2012), p. 613–626.

[151] P. K. Srivastava et al. « A comparison of performance of plant miRNA target prediction tools and the characterization of features for genome-wide target prediction ». en. In : *BMC Genomics* 15.1 (2014), p. 348.

[152] A. Stark et al. « Identification of Drosophila MicroRNA Targets ». In : *PLoS Biol* 1.3 (2003), e60.

[153] G. Stefani et F. J. Slack. « Small non-coding RNAs in animal development ». en. In : *Nature Reviews Molecular Cell Biology* 9.3 (2008), p. 219–230.

[154] A. B. Stergachis et al. « Exonic transcription factor binding directs codon choice and impacts protein evolution ». In : *Science (New York, N.Y.)* 342.6164 (2013), p. 1367–1372.

[155] Y.-H. Sun et al. « Computational Prediction of Plant miRNA Targets ». In : *RNAi and Plant Gene Function Analysis*. Sous la dir. de H. Kodama et A. Komamine. Methods in Molecular Biology 744. 2011, p. 175–186.

[156] R. Sunkar et J.-K. Zhu. « Novel and Stress-Regulated MicroRNAs and Other Small RNAs from Arabidopsis ». In : *The Plant Cell* 16.8 (2004), p. 2001–2019.

[157] R. Sunkar et al. « Small RNAs as big players in plant abiotic stress responses and nutrient deprivation ». In : *Trends in Plant Science* 12.7 (2007), p. 301–309.

[158] Ö. Svensson, L. Arvestad et J. Lagergren. « Genome-Wide Survey for Biologically Functional Pseudogenes ». In : *PLoS Computational Biology* 2.5 (2006).

[159] Y. Tarutani et al. « Trans-acting small RNA determines dominance relationships in Brassica self-incompatibility ». In : *Nature* 466.7309 (2010), p. 983–986.

[160] I. Tinoco et al. « Improved Estimation of Secondary Structure in Ribonucleic Acids ». In : *Nature* 246.150 (1973), p. 40–41.

[161] S. Tonegawa. « Somatic generation of antibody diversity ». In : *Nature* 302.5909 (1983), p. 575–581.

[162] B. J. Tucker et R. R. Breaker. « Riboswitches as versatile gene control elements ». In : *Current Opinion in Structural Biology*. Sequences and topology/Nucleic acids 15.3 (2005), p. 342–348.

[163] E. Ukkonen. « Constructing suffix trees on-line in linear time ». In : *Proc. Information Processing 92*. Sous la dir. d'Elsevier. T. 1. 1992, p. 484–492.

[164] E. Ukkonen. « Finding approximate patterns in strings ». In : *Journal of Algorithms* 6.1 (1985), p. 132–137.

[165] A. V. Uzilov, J. M. Keegan et D. H. Mathews. « Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change ». In : *BMC Bioinformatics* 7 (2006), p. 173.

[166] T. K. Vintsyuk. « Speech discrimination by dynamic programming ». en. In : *Cybernetics* 4.1 (1968), p. 52–57.

[167] O. Voinnet. « Origin, Biogenesis, and Activity of Plant MicroRNAs ». In : *Cell* 136.4 (2009), p. 669–687.

[168] C. Vroland, M. Salson et H. Touzet. « Lossless seeds for searching short patterns with high error rates ». In : *Proc. of International Workshop On Combinatorial Algorithms (IWOCA)*. T. 8986. Lecture Notes in Computer Science. 2014, p. 364–375.

[169] C. Vroland et al. « Approximate search of short patterns with high error rates using the 01*0 lossless seeds ». In : *Journal of Discrete Algorithms* (2016).

[170] R. A. Wagner et M. J. Fischer. « The String-to-String Correction Problem ». In : *J. ACM* 21.1 (1974), p. 168–173.

[171] D. Wang et al. « Induction of Protein Secretory Pathway Is Required for Systemic Acquired Resistance ». In : *Science* 308.5724 (2005), p. 1036–1040.

[172] T. Warnecke, N. N. Batada et L. D. Hurst. « The Impact of the Nucleosome Code on Protein-Coding Sequence Evolution in Yeast ». In : *PLOS Genet* 4.11 (2008), e1000250.

[173] D. Weese, M. Holtgrewe et K. Reinert. « RazerS 3 : Faster, fully sensitive read mapping ». en. In : *Bioinformatics* 28.20 (2012), p. 2592–2599.

[174] P. Weiner. « Linear pattern matching algorithm ». In : *14th IEEE Symp. on Switching and Automata Theory*. 1973, p. 1–11.

[175] B. Wightman, I. Ha et G. Ruvkun. « Posttranscriptional regulation of the heterochronic gene lin-14 by lin-4 mediates temporal pattern formation in C. elegans ». In : *Cell* 75.5 (1993), p. 855–862.

[176] C. Workman et A. Krogh. « No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. » In : *Nucleic Acids Research* 27.24 (1999), p. 4816–4822.

[177] L. Wu et al. « DNA Methylation Mediated by a MicroRNA Pathway ». In : *Molecular Cell* 38.3 (2010), p. 465–475.

[178] F. Xie et B. Zhang. « Target-align : a tool for plant microRNA target identification ». In : *Bioinformatics* 26.23 (2010), p. 3002–3003.

[179] Z. Xie, K. D. Kasschau et J. C. Carrington. « Negative Feedback Regulation of Dicer-Like1 in Arabidopsis by microRNA-Guided mRNA Degradation ». In : *Current Biology* 13.9 (2003), p. 784–789.

[180] Z. Xie et al. « Expression of Arabidopsis MIRNA genes ». In : *Plant Physiology* 138.4 (2005), p. 2145–2154.

[181] K. Xing et X. He. « Reassessing the "hypothesis of protein evolution ». In : *Molecular Biology and Evolution* (2015), msu409.

[182] Y. Zhang. « miRU : an automated plant miRNA target prediction server ». In : *Nucleic Acids Research* 33 (suppl 2 2005), W701–W704.

# Table des matières

**Algorithmique pour la recherche de motifs approchée et application à la recherche de cibles de microARN**


**Résumé**

La recherche de motifs approchée consiste à identifier les occurrences d'un motif modulo une certaine distance au sein d'un texte. Ce problème trouve de nombreuses applications en bio-informatique pour l'analyse de séquences biologiques. Par exemple, les microARN sont des petits ARN qui régulent l'expression des gènes par reconnaissance d'un motif similaire. Comprendre le mode d'action des microARN demande de pouvoir localiser de courts motifs, environ 21 nucléotides, comprenant jusqu'à 3 ou 4 erreurs dans un texte de l'ordre de $10^8$ à $10^9$ nucléotides, représentant un génome. Dans cette thèse, nous proposons un algorithme efficace pour la recherche de motifs approchée, qui se base sur la définition d'un nouveau type de graines avec erreurs, les graines 01*0, et qui exploite une structure d'index compressée, le FM-index. Cet algorithme a été mis en œuvre dans un logiciel librement disponible, appelé Bwolo. Nous démontrons expérimentalement l'avantage de cette approche en nous comparant à l'état de l'art des outils existants. Nous montrons également comment utiliser Bwolo pour mettre en place une analyse originale sur l'étude de la distribution des cibles potentielles de miARN dans deux génomes de plantes, *Arabidopsis thaliana* et *Arabidopsis lyrata*.



**Mots clés :** bioinformatique, algorithmique du texte, recherche de motifs approchée, microARN, *arabidopsis*, régulation de gène


---


**Abstract**

Approximate string matching consists in identifying the occurrences of a motif within a text, modulo a given distance. This problem has many applications in bioinformatics for the analysis of biological sequences. For instance, microRNAs are short RNA molecules regulating the expression of genes by specific recognition of their sequence motif on the target gene. Understanding the mode of action of microRNAs requires the ability to identify short motifs, around 21 nucleotides in size, comprising up to 3-4 errors in a text whose size is in the order of $10^8$-$10^9$, representing a genome. In this thesis, I have proposed an efficient algorithm for the approximate search of short motifs. This algorithm is based on a new type of seeds containing errors, the 01*0 seeds, and uses a compressed index structure, the FM-index. I have implemented this algorithm in a freely available software, Bwolo. I demonstrate experimentally the advantage of this approach and compare it to the state of the art of existing tools. I also show how Bwolo can be used and have set up an original study on the distribution of potential miRNA target sites in two plant genomes, *Arabidopsis thaliana* and *Arabidopsis lyrata*.



**Keywords:** computational biology, text algorithms, approximate string matching, microRNA, *arabidopsis*, gene regulation


---


**CRIStAL – UMR CNRS 9189 Université Lille 1**
Cité scientifique - Bâtiment M3 extension – Avenue Carl Gauss – 59655 Villeneuve d'Ascq Cédex – France