Prédiction de structures secondaires d'ARN et de complexes d'ARN avec pseudonoeuds - Approches basées sur la programmation mathématique multi-objectif

Audrey Legendre

université PARIS-SACLAY

NNT : 2019SACLE031

Thèse de doctorat

Prédiction de structures secondaires d'ARN et de complexes d'ARN avec pseudonoeuds - Approches basées sur la programmation mathématique multi-objectif

Thèse de doctorat de l'Université Paris-Saclay
préparée à Université d'Evry Val d'Essonne

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Evry, le 9 décembre 2019, par

AUDREY LEGENDRE

Composition du Jury :

Peter Clote Professeur, Boston College	Rapporteur
Jérôme Waldispühl Professeur associé, School of Computer Science McGill University	Rapporteur
Alain Denise Professeur, Université Paris-Sud / Paris-Saclay, LRI et I2BC	Président
Patrice Perny Professeur, Sorbonne Université, LIP 6	Examinateur
Yann Ponty Chercheur CNRS, École polytechnique, LIX	Examinateur
Bruno Sargueil Directeur de Recherche CNRS, Université Paris Descartes, CiTCoM	Examinateur
Fariza Tahi Professeur, Univ Evry, Université Paris-Saclay, IBISC	Directrice de thèse
Eric Angel Professeur, Univ Evry, Université Paris-Saclay, IBISC	Co-encadrant de thèse


Mikael Falconnet
Phuong C.
Louis B.
Farida Z.
Gwen
Ying L.
Youri C.
Tiphaine T.
Mikael Falconnet
Muriel B.
Valentin B.
Jean-Christophe J.
Julien B.
Yanting H.
Rodrigue B.
Cristel D.S.C.
Ahmed S.
Nicolas R.
Peipei Z.
Lila V.
Clément B.
Eliott B.C.
Johan A.
Sébastien M.
L3 GBI
Neziha S.
Ilana L.
Ludovic I.
Julien B.
Stanislas H.
Jérémie P.
Méryl V. N.
Hagop K.
Peipei Z.
Victoria B.
Ahmed S.
Johan A.
Annojan K.
Alexandre P.
Nicolas B.
Tiphaine T.
Eric A.
Christine Vassiliadis
Annojan K.
Johan A.
Jean-Christophe J.
Cristel D.S.C.
Lila V.
Quentin P.
Thérèse Malliavin
Tina I.
Ahmed S.
Laurent P.
Yantong L.
Nicolas C.
Eliott B.C.
Nicolas R.
Victoria B. Sébastien M.
Yves-Stan L.C.
Christelle Duhem
Bastien L.
Jean-Marc D.
Christine L.
Célia B.
Deborah M.
Gwen
Fabien J.
Noémie O.D.C.
Kuba S.
Nicolas B.
Tang N.
Gilles B.
Quentin P.
Yantong L.
Deborah M.
Hagop K.
Ludovic I.
Maxime A.
Neziha S.
Laurent P.
Yanting H.
Méryl V. N.
AROBAS
Fariza T.
Hanane O.
Valentin B.
Maxime A.
Sharmili R.
Nesrine B.
Thomas B.
Kuba S.
Nicolas C.
Nesrine B.
Ying L.
Ilana L.
Jean-Marc D.
Thérèse Malliavin
Sabrine L.
Fabien J.
Nicolas C.
Alain L.
Nicolas B.
Célia B.
Sabrine L.
Jérémie P.
Muriel B.
Farida Z.
Alexandre P.
Clément B.
Sharmili R.
Tang N.
Stanislas H.
Hanane O.
Rodrigue B.
Youri C.
Thomas B.

Table des matières

Liste des abréviations **iv**

Table des figures **v**

Liste des tableaux **xi**

Liste des algorithmes **xiii**

Introduction **xiv**

1 Définitions et contexte **5**
1.1 Biologie moléculaire 5
1.1.1 La cellule, hôte de l'information génétique 5
1.1.2 Les ARN : rôles et fonctions 6
1.1.3 La structure secondaire des ARN 9
1.1.4 Les pseudonœuds 12
1.1.5 Les complexes d'ARN 13
1.1.6 Quelques méthodes expérimentales de détermination des structures d'ARN 18
1.2 Algorithmique et optimisation combinatoire 22
1.2.1 La complexité 22
1.2.2 Les graphes 23
1.2.3 Optimisation combinatoire 25
1.3 Bioinformatique des ARN 28
1.3.1 Approches bioinformatiques pour la prédiction de structures secondaires des ARN 28
1.3.2 L'approche thermodynamique 29
1.3.3 Le modèle MFE (*Minimum Free Energy*) 30
1.3.4 Le modèle MEA (*Maximum Expected Accuracy*) 31
1.4 Définition des métriques 32

2 Prédiction de structures secondaires d'ARN avec pseudonœuds **35**
2.1 État de l'art 36
2.1.1 Méthodes et outils ne prédisant pas les pseudonœuds . . 37
2.1.2 Méthodes et outils prédisant les pseudonœuds 40
2.2 Notre méthode : BiokoP 45

2.2.1 Un programme linéaire bi-objectif combinant les modèles MEA et MFE . . . 45
2.2.2 Nouvelle méthode de résolution de programmes bi-objectifs 49
2.2.3 Outil BiokoP . . . 52
2.3 Résultats . . . 52
2.3.1 Jeux de données . . . 52
2.3.2 Comparaison des modèles MFE et MEA . . . 54
2.3.3 Évaluation de BiokoP . . . 54
2.3.4 Comparaison de BiokoP avec la littérature . . . 57
2.3.5 Temps d'exécution . . . 63
2.4 Discussion . . . 64

3 Prédiction de structures secondaires de complexes d'ARN 67
3.1 État de l'art . . . 68
3.1.1 Prédiction d'interaction ARN-ARN . . . 69
3.1.2 Prédiction de complexes d'ARN . . . 69
3.2 Notre méthode : RCPred . . . 71
3.2.1 Un programme linéaire de prédiction d'appariements de complexes d'ARN . . . 72
3.2.2 Approche modulaire et interactive de prédiction de structures de complexes d'ARN . . . 74
3.2.3 Modélisation par une clique pondérée . . . 75
3.2.4 Résolution exacte du problème de la clique pondérée . . . 77
3.2.5 Résolution approchée du problème de la clique pondérée : utilisation d'une heuristique . . . 78
3.2.6 Outil RCPred . . . 80
3.3 Résultats . . . 81
3.3.1 Jeu de données . . . 81
3.3.2 Évaluation de la performance de l'heuristique . . . 82
3.3.3 Qualité des entrées de RCPred . . . 84
3.3.4 Évaluation de RCPred . . . 85
3.3.5 Comparaison de RCPred avec la littérature . . . 90
3.4 Discussion . . . 92

4 Prédiction interactive de complexes d'ARN 95
4.1 État de l'art . . . 97
4.1.1 Interactivité via des contraintes fortes . . . 98
4.1.2 Interactivité via des contraintes faibles . . . 99
4.1.3 Classification de structures secondaires d'ARN . . . 103
4.2 Notre méthode : C-RCPred . . . 105
4.2.1 Un problème de clique multi-objectif . . . 105
4.2.2 Programme linéaire multi-objectif . . . 110
4.2.3 Heuristique multi-critère de recherche de clique . . . 112
4.2.4 Classification des structures secondaires des complexes . 113
4.2.5 Outil C-RCPred . . . 116
4.3 Résultats . . . 118
4.3.1 Jeu de données . . . 119
4.3.2 Évaluation des nouvelles fonctionnalités . . . 119

4.3.3 Évaluation des nouveaux objectifs 124
4.4 Discussion 138

5 Résolution de programmes linéaires en nombres entiers bi-objectifs 141
5.1 État de l'art 142
5.1.1 Obtention de courbes de Pareto exactes 142
5.1.2 Obtention de courbes de Pareto approchées 143
5.1.3 Obtention des k meilleures solutions d'un programme linéaire mono-critère 144
5.2 Notre méthode : Algorithmes et preuves 145
5.2.1 La méthode *ϵ-constraint* 145
5.2.2 Notre méthode de résolution d'un problème bi-objectif : FindKBestParetoSets 146
5.2.3 Preuve 148
5.2.4 Parallélisation de l'algorithme 157
5.2.5 Trouver les k meilleurs ensembles de Pareto parmi un ensemble de points 163
5.2.6 Algorithme approché pour trouver les k meilleurs ensembles de Pareto 165
5.3 Discussion 169

Conclusion et perspectives 170

Bibliographie 177

Liste des abréviations

— A : Adénine.
— ADN : Acide DésoxyriboNucléique.
— ADNc : ADN codant.
— ARN : Acide RiboNucléique.
— ARNm : ARN messager.
— ARNr : ARN ribosomique.
— ARNt : ARN de transfert.
— BLS : *Breakout Local Search*.
— C : Cytosine.
— CMCT : *N-Cyclohexyl-N'-(2-morpholinoethyl)carbodiimide metho-p-toluenesulfonate*.
— DMS : Diméthyle Sulfate.
— FPTAS : *Fully Polynomial-Time Approximation Scheme*.
— G : Guanosine.
— IRES : *Internal Ribosome Entry Site*.
— MAP-seq : *Multiplexed Accessibility Probing sequencing method*.
— MCC : *Matthews Correlation Coefficient*.
— MEA : *Maximum Expected Accuracy*.
— MFE : *Minimum Free Energy*.
— miARN : Micro-ARN.
— NGS : *Next Generation Sequencing*.
— pARNi : petit ARN interférent.
— PARS : *parallel analysis of RNA structure*.
— PCR : *Polymerase chain reaction*.
— PLNE : Programme Linéaire en Nombre Entier.
— PPV : *Positive Predictive Value*.
— pRNA : *prohead RNA*.
— PTAS : *polynomial-time approximation scheme*.
— RMN : Résonance Magnétique Nucléaire.
— SHAPE : *Selective 2'-Hydroxyl Acylation analyzed by Primer Extension*.
— U : Uracile.

Table des figures

1.1 La structure primaire de l'ARN et de l'ADN. 6
1.2 Familles des ARN non-codants. 7
1.3 Les mécanismes de transcription et de traduction. 8
1.4 Les bases azotées. 9
1.5 Les appariements canoniques de l'ARN : G-C et A-U et l'appariement wobble : G-U. 10
1.6 Exemple d'appariements non-canoniques. 10
1.7 Les motifs de la structure secondaire de l'ARN. 11
1.8 Représentation des IRES chez le virus du sida. 11
1.9 Les différents types de pseudonœuds. 12
1.10 Pseudonœuds de profondeur 2 et 3. 13
1.11 Les motifs de la structure secondaire résultants de l'interaction de deux ARN. 14
1.12 Épissage d'un ARN. 14
1.13 Structures du splicéosome. 15
1.14 Structures secondaires et structures 3D des ARN des deux sous-unités composant le ribosome de la levure. 16
1.15 Structure secondaire et modèles en trois dimensions de l'hexamère d'ARN servant au transport du génome du bactériophage $\phi 29$ dans une cellule durant l'infection. 17
1.16 Structure d'un complexe d'ARN synthétique formant un carré. . 18
1.17 Site d'interaction entre différents réactifs et l'ARN lors de méthodes expérimentales de détermination de la structure secondaire. 19
1.18 Représentation des classes de problèmes dans la théorie de la complexité. 22
1.19 Illustration des graphes de cercle sur deux ARN formant une interaction. 25
1.20 Illustration de l'ensemble et du front de Pareto pour un problème bi-objectif où les deux objectifs f_1 et f_2 doivent être minimisés. . 27
1.21 Illustration des notions de solutions supportées et non supportées pour un problème bi-objectif où les deux objectifs f_1 et f_2 doivent être minimisés. 28
1.22 Illustration des 3 ensembles de Pareto pour un problème bi-objectif où les deux objectifs f_1 et f_2 doivent être minimisés. . . 29
1.23 Calcul de l'énergie libre d'une séquence $i\,j$ selon l'algorithme de Nussinov. 30

2.1 Classes de pseudonœuds prédites par les algorithmes de prédiction de structures secondaires 37
2.2 Illustration des contraintes 2.4 (A), 2.5 (B) et 2.6 (C). 48
2.3 Interface du webserver de l'outil BiokoP. 53
2.4 Distribution des structures réelles trouvées par Mod1^{so} et Mod2^{so} sur le jeu de données $pk168$. 54
2.5 Distribution des structures réelles trouvées par BiokoP, Mod1^{so} et Mod2^{so}. 56
2.6 F_1-scores moyens pondérés obtenus sur le jeu de données $B_{\text{nœud}}$ par BiokoP, pKiss, McGenus et IPknot, en fonction du nombre de solutions (NbSol). 59
2.7 F_1-scores des solutions optimales obtenues par BiokoP comparé à pKiss, McGenus, IPknot, MC-Fold, MC-Flashfold et RNAsubopt sur le jeu de données $B_{\text{nœud}}$. 60
2.8 F_1-scores moyens pondérés de BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold et IPknot, en fonction du type de pseudonœuds et du nombre de solutions retournées (NbSol). 61
2.9 F_1-scores moyens pondérés obtenus par BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold, IPknot et RNAsubopt sur le jeu de données $B_{\text{sans nœud}}$, en fonction du nombre de solutions (NbSol). 62
2.10 Prédictions de structures secondaires avec BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold, IPknot et RNAsubopt pour le pseudonœud PK4 de l'ARN télomérase et pour un pseudonœud en forme de marteau. 63

3.1 Définitions des motifs d'un complexe d'ARN. 68
3.2 Principe général de notre méthode RCPred. 75
3.3 Ensembles utilisés dans l'heuristique *breakout local search*. . . . 79
3.4 RCPred sur la plateforme EvryRNA. 82
3.5 Performance de l'heuristique : poids de la clique optimale. . . . 83
3.6 Performance de l'heuristique : temps d'éxecution. 84
3.7 MCC des structures et interactions d'entrée de RCPred pour chaque complexe de C_{multi}. 86
3.8 MCC des structures d'entrée de RCPred pour chaque complexe de C_{multi} suivant l'outil utilisé. 87
3.9 MCC moyens sur 10 exécutions des 10 premières structures secondaires retournées par RCPred sur le jeu de données C_{multi}. . 88
3.10 MCC moyens sur 10 itérations des 10 premières structures secondaires retournées par RCPred sur le jeu de données C_{multi} en fonction des structures secondaires utilisées en entrée. 89
3.11 MCC obtenus par RCPred, NanoFolder, NUPACK et MultiRNAfold sur chaque complexe de C_{multi}. 91

4.1 Principe de l'outil C-RCPred. 97
4.2 Illustration des contraintes utilisateurs de motifs possibles pour un complexe d'ARN. 106
4.3 Nombre de conflits autorisés en fonction de T et du seuil. 107

4.4 Illustration de la relation $R_{r,d}$ du noyau développé par Costa et De Grave, pour $r = 1$ et $d = 5$. 114
4.5 Capture d'écran du webserver de C-RCPred. 117
4.6 MCC moyens et maximaux sur les 10^e structures prédites par C-RCPred, en fonction du seuil de compatibilité sur le jeu de données C_{multi}. 120
4.7 Structure de référence du complexe PDB_00851. 121
4.8 Structure secondaire d'un exemple de pseudonœud de niveau 4 grâce à deux séquences d'ARN du complexe PDB_00851. 122
4.9 Carte thermique d'une matrice du noyau NSPDK de taille 120×120. 123
4.10 Nombres moyens de structures secondaires obtenues par C-RCPred avec ou sans classification sur le jeu de données C_{multi}. 124
4.11 MCC moyens de structures secondaires obtenues par C-RCPred sans classification (A) ou avec (B) classification sur le jeu de données C_{multi}. 125
4.12 Structure secondaire de référence du complexe PDB_01165. . . 125
4.13 MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs a et b sur le complexe PDB_01165. . . . 127
4.14 MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs a et b et c sur le complexe PDB_01165. . 128
4.15 MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs a et b et c, avec différents indices de confiance, sur le complexe PDB_01165. 129
4.16 MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs d et e sur le complexe PDB_01165. . . . 130
4.17 Exemples de structures obtenues par C-RCPred avec les contraintes utilisateurs d (A) et e (B) sur le complexe PDB_01165. . . 130
4.18 Structures de référence des complexes B et C du splicéosome humain colorées selon les données structurales de PARS. 131
4.19 Structure de référence du complexe d'ARN carré colorée selon les données structurales de SHAPE. 132
4.20 MCC des interactions et des structures secondaires utilisées en entrée de C-RCPred pour les complexe B et C. 133
4.21 MCC des interactions et des structures secondaires utilisées en entrée de C-RCPred pour le complexe carré. 135
4.22 Pipeline d'évaluation pour les complexes B, C et carré. 136
4.23 MCC des structures prédites pour les complexes B, C et carré, par les outils C-RCPred, NanoFolder, MultiRNAfold et NUPACK. . . 137

5.1 Représentation par un arbre d'un exemple d'exécution de l'algorithme FindKbestParetoSets 151
5.2 Parentalité entre les solutions de l'algorithme FindKbestParetoSets 151
5.3 Illustration de la propriété 5.2.6. 152
5.4 Cas d'un alignement vertical de solutions. 153
5.5 Mise à jour de l'ensemble F lors d'un appel vers le haut. 155
5.6 Illustration d'une exécution de l'algorithme FindKbestParetoSets (algorithme 7) parallélisé. 158

5.7 Les différentes étapes de génération d'un arbre binaire aléatoire. 159
5.8 Évolution du coefficient d'accélération d'un arbre d'exécution en fonction de la probabilité p. 163
5.9 Exemple d'exécution non parallélisable de l'algorithme FindKbestParetoSets 163
5.10 Illustration du principe de l'algorithme 9 pour trouver des solutions aux programmes P_1 et P_2. 166

Liste des tableaux

1.1 Paramètres énergétiques d'empilement d'appariements. 31
1.2 Paramètres énergétiques de boucles. 31

2.1 Résumé de l'état de l'art des outils de prédiction de structures secondaires d'ARN sans pseudonœuds. Les outils en gris ne sont plus ou pas encore disponible, ou ils sont obsolètes. 38
2.2 Résumé de l'état de l'art des outils de prédiction de structures secondaires d'ARN avec pseudonœuds. 41
2.3 Paramètres énergétiques des paires d'appariements utilisés dans le programme linéaire de Poolsap. 47
2.4 Distribution des structures réelles trouvées par BiokoP en fonction du nombre d'ensembles de Pareto générés sur le jeu de données $B_{\text{nœud}}$. 55
2.5 Moyennes pondérées en fonction du nombre de solutions (NbSol) des sensitivités obtenues par BiokoP, pKiss, McGenus, MC-Fold et MC-Flashfold sur le jeu de données $B_{\text{nœud}}$. 58
2.6 Moyennes pondérées en fonction du nombre de solutions (NbSol) des PPV obtenus par BiokoP, pKiss, McGenus, MC-Fold et MC-Flashfold sur le jeu de données $B_{\text{nœud}}$. 58

3.1 Résumé de l'état de l'art des outils de prédiction de structures secondaires d'un duplexe d'ARN. 70
3.2 Résumé de l'état de l'art des outils de prédiction de structures secondaires d'un complexe d'ARN. 71
3.3 Sensitivité, PPV, MCC et F_1-score moyens sur C_{multi} obtenus par RCPred, NanoFolder, NUPACK et MultiRNAFold. 92

4.1 Récapitulatif des outils de prédiction de structures secondaires prenant en compte des contraintes fortes. 98
4.2 Récapitulatif des outils de prédiction de structures secondaires prenant en compte des données structurales. 100
4.3 Récapitulatif des méthodes et outils de classification de structures secondaires d'ARN et de complexe d'ARN. 103
4.4 MCC, sensitivités et PVV, moyens et maximaux, des 10 premières structures de C-RCPred en fonction de la compatibilité sur le jeu de données C_{multi}. 120

4.5 Contraintes utilisateurs utilisées avec C-RCPred sur le complexe PDB_01165. 126

5.1 Règles lors de la génération d'un arbre représentant l'exécution de l'algorithme FindKbestParetoSets. 162

Liste des Algorithmes

1 FindKParetoSets(k) 51
1 Procédure LocalPareto(λ^-, λ^+, F) 51

2 Coloration optimale d'un graphe d'intervalles. 110
3 Classification spectrale. 115
4 k-moyennes. 115
5 Méthode *eigengap* donnant un nombre de classes d'une matrice de données. 116

6 La méthode ϵ-*constraint* appliquée à un problème bi-objectif . . 146
7 FindKbestParetoSets 148
2 Procédure LocalPareto(λ^-, λ^+, F) 149
8 FindKbestParetoSetsScatterPlot 164
9 Méthode ϵ-*constraint* bi-objectif $(1+\alpha)$-approchée. 167
10 $(1+\alpha)$-approché FindKbestParetoSets 168
3 Procédure $1+\alpha$-approché LocalPareto(λ^-, λ^+, F) 169

Introduction

Les ARN exercent de nombreux rôles au sein de la cellule [254]. Les ARN codants, ou ARN messagers (ARNm), servent de support de traduction pour la synthèse des protéines, tandis que les ARN non-codants servent en grande majorité à la régulation de l'expression des gènes et à la traduction. Les ARN non-codants peuvent également exercer une activité catalytique, ils sont alors appelés *ribozymes*. De plus, en biologie de synthèse, grâce à la flexibilité de leur structure, les ARN sont utilisés dans des but thérapeutiques ou industriels [13].

Les ARN n'ont pas une structure linéaire en double hélice comme l'ADN. En effet, le plus souvent, l'ARN est sous forme d'un simple brin. Il peut alors se replier sur lui-même en une structure complexe à la manière des protéines. Dans un premier temps, lorsque la séquence nucléotidique, appelée la *structure primaire*, n'est pas encore repliée, des liaisons entre paires de nucléotides se forment ; apparaissent alors des motifs : des boucles, des tiges et des motifs particuliers, appelés *pseudonœuds*. Ces différents motifs forment la *structure secondaire*. Enfin, d'autres liaisons plus faibles se forment entre les nucléotides (pouvant impliquer plus de deux nucléotides), donnant ainsi sa structure 3D à l'ARN : c'est la *structure tertiaire*.

La structure d'une molécule biologique aide à la détermination de sa fonction. Cela peut être une fonction mécanique, comme les IRES (*Internal Ribosome Entry Site*) qui sont une suite de tige-boucles en amont d'un ARN messager et qui permettent le recrutement du ribosome pour la traduction [83]. Cela peut être une fonction catalytique, par exemple le ribosome [53] ou le splicéosome [234] qui sont des complexes d'ARN et de protéines dont l'activité catalytique est réalisée par les ARN.

Les nombreux rôles de l'ARN dans la cellule, ainsi que son utilisation en biologie de synthèse [57], font que la détermination précise de sa structure est une question primordiale en recherche fondamentale et applicative. À l'heure actuelle, la détermination de la structure par expérimentation peut se réaliser par des techniques comme la résonance magnétique nucléaire (RMN) [171] ou la cristallographie par rayon X [77]. Cependant ces techniques s'avèrent être difficiles à mettre en œuvre, longues et coûteuses. C'est pourquoi un large domaine de la bioinformatique est dédié à la prédiction des structures d'ARN. La prédiction en trois dimensions d'une molécule est un problème très difficile car il est basé sur la prédiction des liaisons atomiques. Une manière de rendre la problématique plus abordable est de s'appuyer sur la structure secondaire

qui elle, est basée sur la prédiction des appariements nucléotidiques. De plus, la structure secondaire est souvent suffisante pour mener des analyses sur les ARN : détermination de la fonction, biologie synthétique, classification, études comparatives, *etc*.

Il existe plusieurs modèles de prédiction bioinformatique de la structure secondaire qui peuvent s'exprimer sous forme de problèmes d'optimisation, visant notamment la minimisation de l'énergie libre (MFE, *Minimum Free Energy*) [256] ou la maximisation de la précision attendue (MEA, *Maximum Expected Accuracy*) [134]. C'est un domaine de la bioinformatique actif et de nombreux outils de prédiction existent. Les méthodes exploitant les informations évolutives d'ARN homologues, appelées *méthodes comparatives*, permettent d'obtenir de bonnes prédictions de structures. Cependant ces informations ne sont pas toujours disponibles. Les modèles thermodynamiques sont, par conséquent, plus largement utilisés, c'est à ces modèles que nous nous sommes intéressés.

Nous nous intéressons également à la prédiction de structures secondaires incluant des pseudonœuds. Les pseudonœuds font perdre la conformation plane de la structure secondaire, c'est pourquoi ils sont parfois considérés comme faisant partie de la structure tertiaire. Ces motifs sont plus difficile à prédire, en effet, dans la plupart des modèles MFE, il n'y a pas de paramètres d'énergie relatifs aux pseudonœuds. De plus, leur prédiction a entraîné pendant un certain temps une complexité algorithmique plus importante [153] et par conséquent ils étaient le plus souvent ignorés lors de la prédiction.

L'environnement des ARN joue également un rôle important dans leur structure. Par nature, l'ARN est moins stable que l'ADN, sa structure peut se modifier facilement. Les ARN peuvent donc se lier à d'autres molécules présentes dans leur environnement comme des ions, des protéines ou encore d'autres ARN. Pour former ces liaisons, un ARN peut se replier en une structure différente de sa structure initiale. C'est par exemple le cas des *riboswitches* [205]. Ainsi, la prédiction de la structure la plus probable d'un ARN n'est souvent pas suffisante. Une approche possible est de faire en sorte que les modèles de prédiction puissent générer un ensemble de structures plutôt qu'une seule. Une alternative est d'essayer de prédire la structure de l'ARN conjointement avec la ou les molécules qui doivent se lier à lui. Nous avons choisi d'utiliser les deux options à la fois. Dans cette thèse nous nous sommes en effet intéressés plus particulièrement à la prédiction de complexes d'ARN et de structures sous-optimales.

Il existe des méthodes expérimentales qui permettent d'obtenir des informations de structure sur un ARN ou un ensemble d'ARN donnés et qui peuvent être utilisées pour la prédiction bioinformatique. Ces méthodes permettent d'obtenir notamment la réactivité d'un nucléotide à un produit chimique, comme le SHAPE [24] ou le DMS [43]. Si un nucléotide a une réactivité élevée c'est qu'il n'est pas engagé dans un appariement, et inversement. L'ajout de ces données expérimentales dans la prédiction de la structure d'ARN permet de guider la prédiction.

Pour intégrer ce type de données dans les algorithmes de prédiction, des contraintes "faibles" sont utilisées. Un algorithme avec des contraintes faibles

doit essayer de les respecter au mieux dans les solutions retournées, tandis qu'il doit obligatoirement respecter des contraintes fortes. Dans cette thèse, nous proposons de prendre en compte des données structurales sous la forme de contraintes faibles. Nous souhaitons également prendre en compte des contraintes utilisateurs, sous la forme de contraintes fortes, pour intégrer des contraintes d'appariements ou de motifs.

Pour prédire des structures secondaires d'ARN, l'approche algorithmique la plus utilisée est la programmation dynamique [259], contrairement la programmation linéaire [167]. Nous nous sommes intéressés dans cette thèse à la programmation mathématique linéaire à cause de sa flexibilité. En effet, elle permet de modéliser facilement la prédiction de tous les types de pseudonœuds, ce qui n'est pas le cas de la programmation dynamique. Cependant, résoudre un programme mathématique linéaire en nombres entiers est exponentiel en temps, ce qui n'est pas praticable sur de grandes instances. Cela nous a conduit à développer des heuristiques originales pour répondre à nos problèmes. Nous nous sommes également intéressés à la programmation linéaire multi-objectif afin de combiner plusieurs modèles de prédiction. En effet, les structures d'ARN prédites variant selon le modèle utilisé, la combinaison de plusieurs modèles permet de tirer parti de chacun d'eux.

Dans cette thèse, nous proposons trois nouvelles méthodes pour la prédiction de structures secondaires d'ARN et de complexes d'ARN avec pseudonœuds :

— La première méthode que nous proposons consiste à trouver des structures secondaires d'ARN avec pseudonœuds, optimales et sous-optimales, en combinant les deux modèles de prédiction MFE et MEA. Cette méthode a donné lieu à un outil appelé BiokoP [122].
— La deuxième méthode consiste à prédire un ensemble de structures secondaires de complexes d'ARN, à partir d'un ensemble de structures secondaires et d'interactions, donnant lieu à l'outil RCPred [123].
— La troisième méthode permettant de prédire, elle aussi, des structures secondaires de complexes d'ARN, mais permet de prendre en compte contraintes utilisateurs et des données structurales. Elle a donné lieu à l'outil C-RCPred.

Les méthodes que nous proposons dans cette thèse sont toutes différentes d'un point de vue algorithmique. En effet, l'outil BiokoP nous a permis de développer un algorithme pour trouver les k meilleurs ensembles de Pareto exacts d'un programme linéaire en nombres entiers bi-objectif, c'est-à-dire des solutions optimales et sous-optimales exactes. Nous avons également développé une version parallélisée et une version approchée de cet algorithme. Les outils RCPred et C-RCPred utilisent, quant à eux, des heuristiques pour résoudre des programmes linéaires en nombres entiers, dont la résolution exacte est exponentielle. Pour RCPred, nous avons adapté une heuristique, permettant de résoudre un programme linéaire mono-objectif, pour qu'elle génère un ensemble de solutions approchées. Le programme linéaire de RCPred est le problème de la clique maximum qui est NP-difficile et difficile à approximer [28, 89]. L'heuristique de RCPred nous a servi de base pour l'outil C-RCPred, où l'heuristique a pour but de résoudre un programme linéaire multi-objectif et

générant un ensemble de Pareto approché.

Ce manuscrit de thèse est composé de cinq chapitres. Le premier chapitre est consacré à des rappels et à des définitions nécessaires à la compréhension du contexte biologique et bioinformatique de nos travaux, ainsi qu'aux différents concepts utilisés dans les méthodes et algorithmes que nous avons développés. Le deuxième chapitre est consacré à notre méthode bi-objectif BiokoP, que nous avons développée pour la prédiction de structures secondaires, incluant les pseudonœuds, d'un ARN donné. Il y est présenté notre méthode bi-objectif combinant les deux modèles de prédiction avec la programmation linéaire, et les résultats de notre méthode sur un ensemble d'ARN connus. Le troisième chapitre est dédié à notre méthode RCPred, développée pour la prédiction de la structure secondaire de complexes d'ARN. Il y est inclus la modélisation du problème sous forme de recherche de cliques contraintes dans un graphe et sa résolution par une heuristique basée sur la recherche locale. Nous présentons également les résultats de notre méthode sur un ensemble de complexes connus. Le quatrième chapitre est consacré à notre méthode C-RCPred, développée pour la prédiction interactive de la structure secondaire de complexes d'ARN, prenant en compte des données structurales et des contraintes utilisateurs. Dans ce chapitre, notre heuristique multi-critère pour résoudre un problème de clique multi-objectif est présentée, ainsi que les résultats de notre méthode. Dans le cinquième chapitre, nous proposons un algorithme original de génération des k meilleurs fronts de Pareto exacts (dont le front de Pareto optimal), ainsi que sa preuve, une parallélisation possible et des algorithmes approchés. Nous présentons également un algorithme résolvant un problème connexe à la génération des k meilleurs fronts de Pareto. Enfin, le dernier chapitre est dédié aux conclusions et perspectives liées à ces méthodes.

1 Définitions et contexte

1.1 Biologie moléculaire

1.1.1 La cellule, hôte de l'information génétique

La *cellule* est la plus petite unité vivante capable de se reproduire. Elle est constituée d'une membrane plasmique qui englobe le cytoplasme dans lequel se trouve le matériel nécessaire à son fonctionnement.

Dans la cellule, le *génome* désigne l'ensemble des acides désoxyribonucléiques (ADN) qui stockent l'information pour générer les composants de la cellule ; le *transcriptome* désigne l'ensemble des acides ribonucléiques (ARN) et le *protéome* désigne l'ensemble des protéines. Les protéines sont à la fois des briques constituant la structure de la cellule et des outils pour son fonctionnement. Elles sont générées par la *transcription* de gènes présents dans l'ADN en *ARN messagers* (ARNm) qui sont eux-mêmes traduits en protéine par le mécanisme de la *traduction*.

Il existe deux grands types de cellules : celles ayant un noyau abritant le matériel génétique, ce sont des cellules *eucaryotes*, et les cellules sans noyau, rassemblant les *procaryotes* et les *archées*. Parmi les organismes eucaryotes on trouve les animaux et les plantes dont les cellules sont assez différentes. Chez les plantes, il y a par exemple une paroi cellulaire qui protège la cellule (c'est le cas aussi chez les procaryotes) et des chloroplastes qui sont responsables de la photosynthèse. Les procaryotes rassemblent tous les organismes bactériens.

Hormis le noyau, on trouve dans la cellule d'autres compartiments appelés *organites* ayant une fonction biologique spécifique. Par exemple les ribosomes sont présents chez tous les organismes, ils sont responsables de la traduction d'ARNm en protéines.

Aujourd'hui, la cellule est largement étudiée par le biais de son génome, c'est le domaine de la génomique. La génomique s'est développée grâce à l'essor des nouvelles technologies de séquençage (NGS) qui permettent de séquencer efficacement et à bas prix un génome entier. Le génome peut alors être annoté et étudié, notamment dans le cadre de la génomique comparative

qui permet d'étudier les différences entre les génomes (entre deux espèces différentes ou entre deux individus sain et malade par exemple). Les NGS ouvrent de nombreuses possibilités d'étude de la cellule à grande échelle. Elles permettent de déterminer le transcriptome à un moment donné du cycle cellulaire ou dans des conditions particulières, par exemple de maladie : c'est le profil d'expression des gènes. Les NGS couplées à des techniques de détermination de structure permettent de déterminer des informations structurales d'un ensemble d'ARN à l'échelle génomique. Certaines de ces techniques seront décrites en section 1.1.6.

1.1.2 Les ARN : rôles et fonctions

Les acides ribonucléiques ou ARN sont des molécules constituées d'une suite de nucléotides. Les nucléotides comprennent un sucre à cinq carbones appelé ribose (d'où le nom d'acide ribonucléique), auquel est lié un groupement tri-phosphate et une base azotée : adénine (A), uracile (U), guanine (G) ou cytosine (C) (voir figure 1.1). Les riboses et les phosphates forment ensemble le *squelette* de l'ARN. La suite de nucléotides forme la séquence : la *structure primaire* de l'ARN.

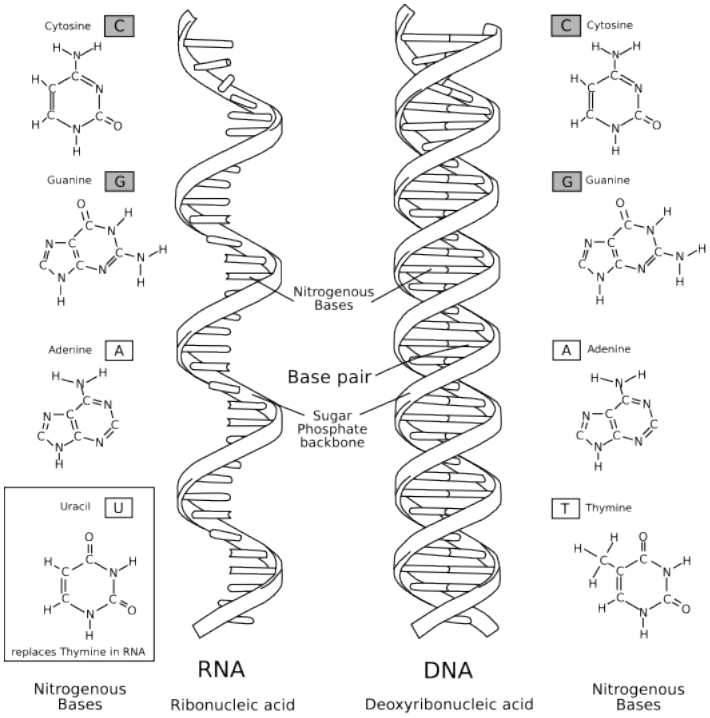

Figure 1.1 : La structure primaire de l'ARN et de l'ADN. (http ://rosalind.info/problems/locations/).

Les ARN sont présents dans les cellules et dans les virus. Chez les virus, l'ARN sert de stockage du génome, au même titre que l'ADN chez les organismes. Les ARN exercent de nombreux rôles au sein de la cellule. Les ARN codants, ou ARNm, servent de supports de traduction pour la synthèse protéomique. Les ARN non-codants (ARNnc) possèdent également de nombreux

rôles au sein de la cellule. Ils peuvent être répartis en de nombreuses grandes familles au vue de leur importante diversité (voir figure 1.2).

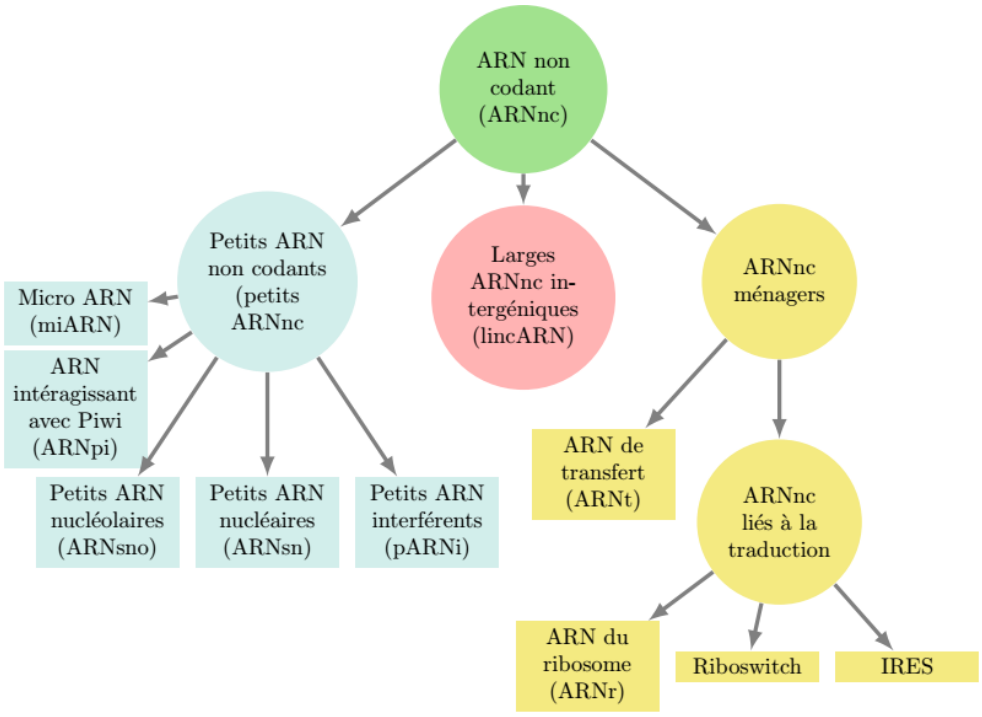

Figure 1.2 : Familles des ARN non-codants.

Les ARNm subissent plusieurs transformations avant d'être traduits en protéines (voir figure 1.3). Une étape importante est notamment l'épissage où un complexe d'ARN, appelé *splicéosome*, extrait certaines parties de l'ARN, appelées introns, pour ne garder que les exons, qui, assemblés et traduits, forment une protéine. Parfois, des exons peuvent être assemblés dans différents ordres (parfois pas tous les exons également), cela permet d'obtenir différentes protéines à partir d'une même transcription d'un gène, c'est ce qui est appelé l'épissage alternatif,

Parmi les ARNnc, les ARN de transfert (ARNt) apportent les acides aminés correspondant au code génétique lors de la traduction. Ils forment alors, avec l'ARN messager et le ribosome, la machinerie permettant la traduction. Le code génétique permet de faire le lien entre des triplets de nucléotides consécutifs composant l'ARN et les acides aminés composant les protéines. Les micro-ARN (miARN) et les petits ARN interférents (pARNi) régulent l'expression de gènes par des mécanismes comme la dégradation d'ARN messagers, la répression ou le renforcement traductionnel. Ces mécanismes sont réalisés notamment grâce à des complexes ARN-protéine. Les ARN de voûte (ARNv) sont des petits ARN faisant partie des complexes ARN-protéines, appelés voûtes, associés aux pores nucléaires chez les eucaryotes. Les ARN ayant une activité catalytique sont appelés *ribozymes*. Le ribosome, le complexe de protéines et d'ARN responsable de la traduction, est un ribozyme. Tout comme la télomérase qui s'occupe d'ajouter des télomères (petites structures d'ADN) aux chromosomes pour que leur longueur soit conservée. La ribonucléase P est également un ribozyme,

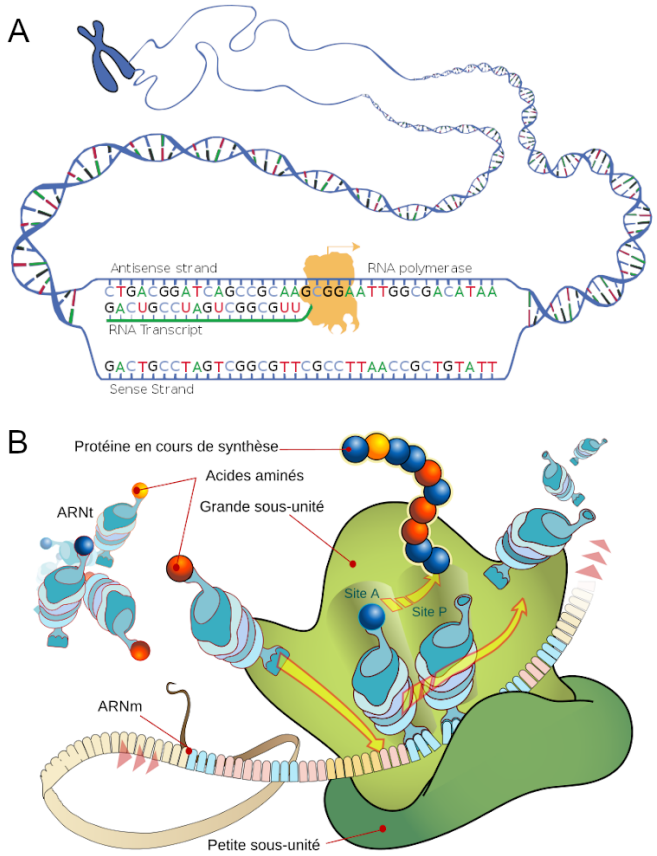

Figure 1.3 : Les mécanismes de transcription (A) et de traduction (B). Sources : National Human Genome Research Institute (A) et Wikimedia, LadyofHats (B).

elle mature les ARNt pour qu'ils soient fonctionnels.

Les ARN sont responsables de nombreux mécanismes et par conséquent peuvent être impliqués dans diverses maladies. La base de données NONCODE 2016 [254] rassemble notamment 527336 longs ARN non-codants de 16 espèces différentes et donne des informations quant à leur lien avec les maladies.

Les ARN peuvent être utilisés comme cibles thérapeutiques [13]. Les ARN sont également utilisés dans un but thérapeutique par la biologie de synthèse. Par exemple, les miARN se fixent à un ARNm complémentaire et stoppent leur traduction ou entraîne leur dégradation par le complexe RISC. Ils peuvent alors être utilisés pour empêcher l'expression de certains gènes, notamment pour contrer le cancer [187]. Les ribozymes eux aussi sont l'objet du développement d'outils thérapeutiques. Un ribozyme a été développé pour cliver le génome du virus du sida [8]. Les aptamères sont des ARN n'ayant aucune fonction catalytique mais qui se lient à d'autres ARN, à de l'ADN ou encore à des protéines. Le médicament Mucagen est basé sur un aptamère [155], appelé pegaptanib sodium, se liant à la protéine VEGF impliquée dans le développement des vaisseaux sanguins. Ce médicament a été développé dans le but de traiter la dégénérescence maculaire liée à l'age (DMLA). Les aptamères peuvent également être utilisés dans le but d'établir des diagnostiques. Par exemple un aptamère déterminant une carence en vitamine B12 a été développé [197]. Les riboswitches sont des ARN qui peuvent adopter deux conformations. Le

changement de conformation est dû à la liaison de l'ARN avec un ligand. Les riboswitches régulent l'expression de gènes et peuvent être utilisés comme cible thérapeutique : des ligands analogues à ceux se liant aux riboswitches sont développés pour perturber les mécanismes de régulation [205].

1.1.3 La structure secondaire des ARN

La structure secondaire des ARN est définie comme l'ensemble des liaisons entre les nucléotides, ou bases azotées. Certaines bases s'apparient préférentiellement avec d'autres, suivant les possibilités de liaisons qui sont offertes, grâce à des liaisons hydrogènes. Les appariements les plus stables sont les appariements *Watson-Crick*, dits aussi *canoniques* : A se lie avec U et G se lie avec C. Les autres types d'appariements sont dits *non-canoniques*. Sur les bases azotées, trois faces permettent de réaliser des liaisons hydrogènes. Ces trois faces sont appelées *Watson-Crick*, *Hoogsteen* ou *C-H* et *ribose* ou *sucre* (voir figure 1.4 A). Le sens dans lequel se trouvent les bases azotées par rapport au ribose, au niveau de la liaison glycosidique, est appelé *trans* ou *cis* (voir figure 1.4 B). Les différents appariements peuvent être qualifiés suivant les faces d'interaction et le sens de la liaison glycosidique ; ainsi, de nombreuses liaisons sont possibles (les figures 1.5 et 1.6 en décrivent une partie). Cependant, les appariements les plus courants sont les plus stables, comme les appariements Watson-Crick cités précédemment. Un autre appariement, un peu moins stable mais pourtant très répandu, est l'appariement G-U, dit *wobble*.

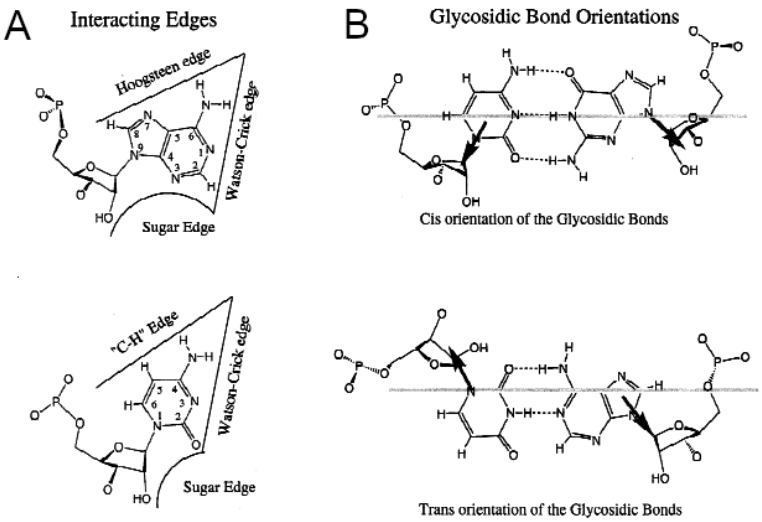

Figure 1.4 : Les bases azotées. A) Les faces d'interactions : en haut les purines (A et G) et en bas les pyrimidines (C et U) [125]. B) Les liaisons glycosidiques : en haut de type *cis* et en bas de type *trans* [125].

L'ensemble de ces appariements, A-U, G-C et G-U, sont représentés forme la *structure secondaire* de l'ARN. Ces appariements font apparaître des motifs : des tiges et des boucles (voir figure 1.7). Les tiges sont des ARN double brin, qui comme l'ADN, vont former une *hélice* en trois dimensions. Une boucle terminant une hélice est appelée *boucle terminale*, tandis que si elle se situe au

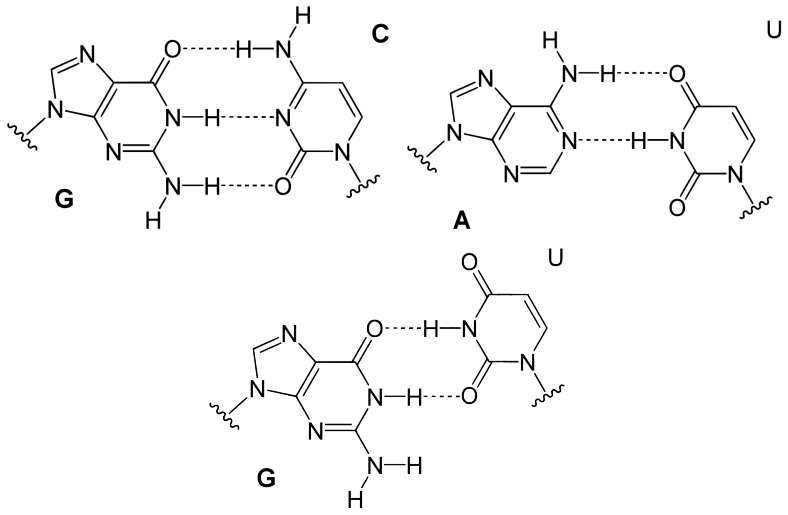

Figure 1.5 : Les appariements canoniques de l'ARN : G-C et A-U et l'appariement wobble : G-U.

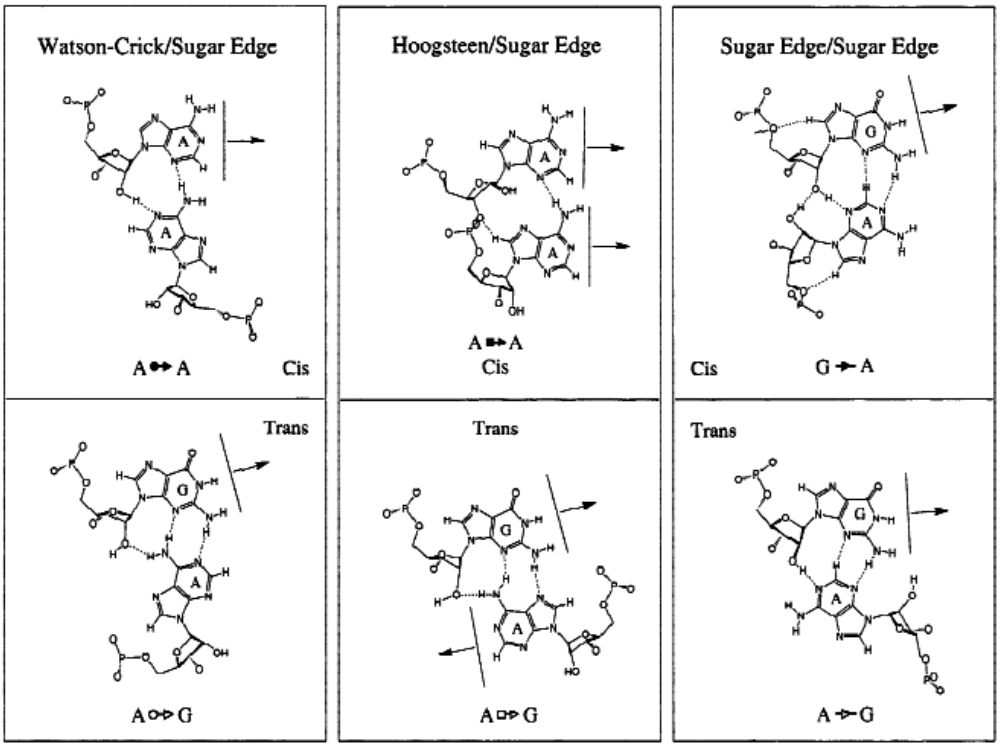

Figure 1.6 : Exemple d'appariements non-canoniques [125].

milieu d'une hélice elle est appelée *boucle interne*. Si dans une boucle interne, les nucléotides non-appariés se situent d'un seul côté de l'hélice, alors elle est appelée *renflement*. Des nucléotides non-appariés qui permettent de rejoindre plusieurs hélices forment une *jonction multiple*. Enfin, quand des nucléotides non-appariés d'une boucle (terminale ou interne) s'apparient avec une autre section de l'ARN, cela forme un *pseudonœud*.

La structure secondaire des ARN est importante pour la fonction de l'ARN. Les *internal ribosome entry site* (IRES) sont une suite de structures tige-boucle

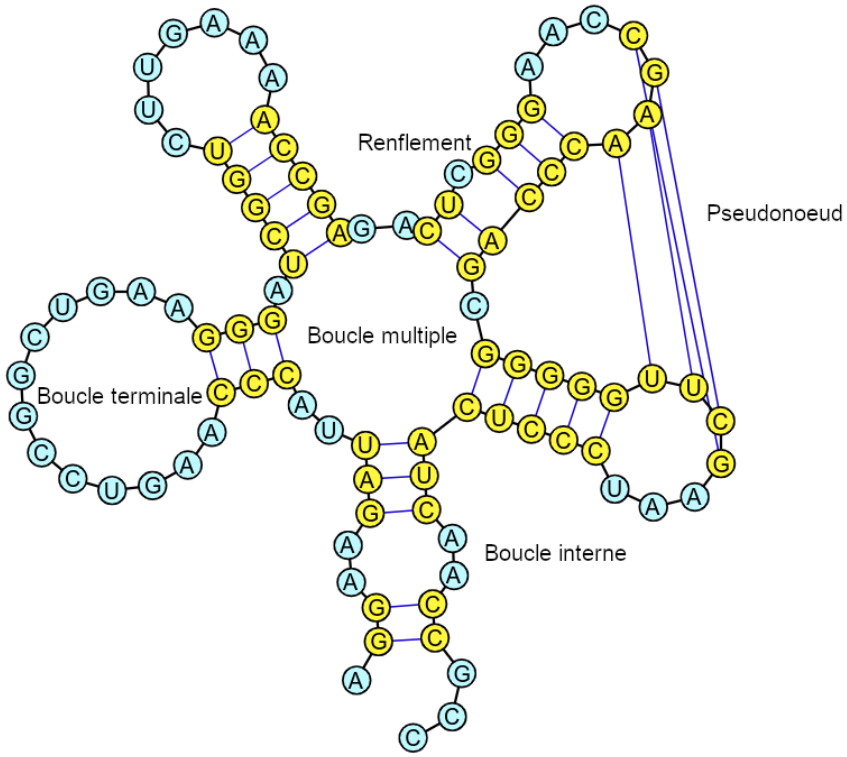

Figure 1.7 : Les motifs de la structure secondaire de l'ARN.

(voir figure 1.8) en amont de gènes chez les virus. Ces structures vont recruter le ribosome pour effectuer la traduction des gènes. Ces structures se présentent par exemple chez le virus du sida [83]. La détection de ces IRES permet de développer de nouveaux médicaments, par exemple le médicament Fuzeon cible l'IRES du HIV [149].

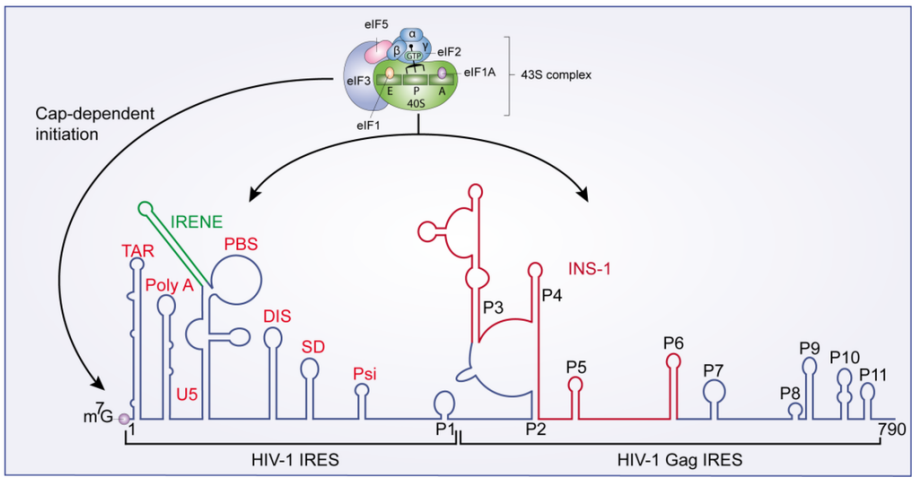

Figure 1.8 : Représentation des IRES chez le virus du sida [83].

Chez les ARN, en plus des liaisons hydrogènes impliquées dans les appariements canoniques, il existe des liaisons hydrogènes responsables de la

structure 3D ou *tertiaire* d'un ARN. Ces liaisons sont plus nombreuses au niveau des nucléotides non-appariés car leurs atomes ne sont pas engagés dans des liaisons canoniques. Les appariements non-canoniques (voir figure 1.6) font partie de ces liaisons.

1.1.4 Les pseudonœuds

Quand des nucléotides non-appariés d'une boucle s'apparient avec une autre section de l'ARN, cela forme un *pseudonœud*. Formellement, lorsque deux appariements (i, j) et (k, l) sont tels que $i < k < j < l$, alors ils forment un pseudonœud. Plusieurs types de pseudonœuds se distinguent. Une classification a été établie lors de la réalisation de la base de données Pseudobase++ [212] qui rassemble des séquences d'ARN dont la structure secondaire inclut au moins un pseudonœud. Cette classification (figure 1.9) est basée sur la topologie des pseudonœuds. Le plus répandu dans les structures connues est le type H. Vient ensuite le type HHH, appelé aussi *kissing hairpin*. Enfin les autres types définissent des pseudonœuds dont l'implantation d'hélices à l'intérieur du pseudonœud diffère.

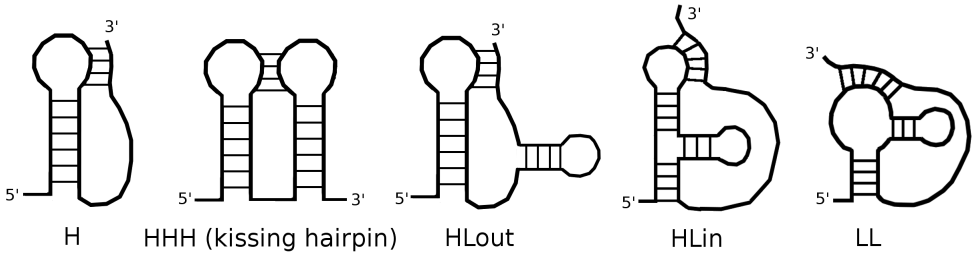

Figure 1.9 : Les différents types de pseudonœuds.

Ces motifs particuliers de la structure secondaire jouent un rôle important dans la structure des ARN. En effet, ils sont retrouvés dans des mécanismes comme le décalage du cadre de lecture lors de la traduction [79] ou chez des ribozymes [82]. Lors d'un décalage du cadre de lecture, un pseudonœud peut se trouver dans le ribosome au niveau de l'ARN messager. Le pseudonœud va empêcher le bon déroulement de la traduction, il sera déplié ou déformé pour que la traduction de l'ARNm continue. Cela va provoquer le décalage du cadre de lecture et l'ARNm sera traduit différemment. Le pseudonœud peut également stopper complètement la traduction sans qu'il n'y ait de décalage du cadre de lecture. Ce mécanisme se retrouve dans le génome du virus du syndrome d'immunodéficience humaine [99], où un pseudonœud, présent dans l'ARNm codant pour la polyprotéine gag-polymérase, peut bloquer la traduction dans 10% des cas. Les pseudonœuds peuvent également se trouver dans la structure de ribozymes, comme par exemple chez la télomérase [82].

Les pseudonœuds sont le résultat d'au moins un croisement entre deux hélices. Il peut arriver qu'il y ait plusieurs croisements, augmentant la complexité d'un pseudonœud. Cette complexité peut être évaluée par la *profondeur* du pseudonœud. Cette propriété est également appelée *niveau* [167] ou numéro de page [42] d'un pseudonœud.

Définition 1.1.1. La *profondeur* d'un pseudonœud correspond à la taille minimum du nombre de sous-ensembles d'hélices d'une structure tel que chaque sous-ensemble ne contienne aucun pseudonœud (voir figure 1.10).

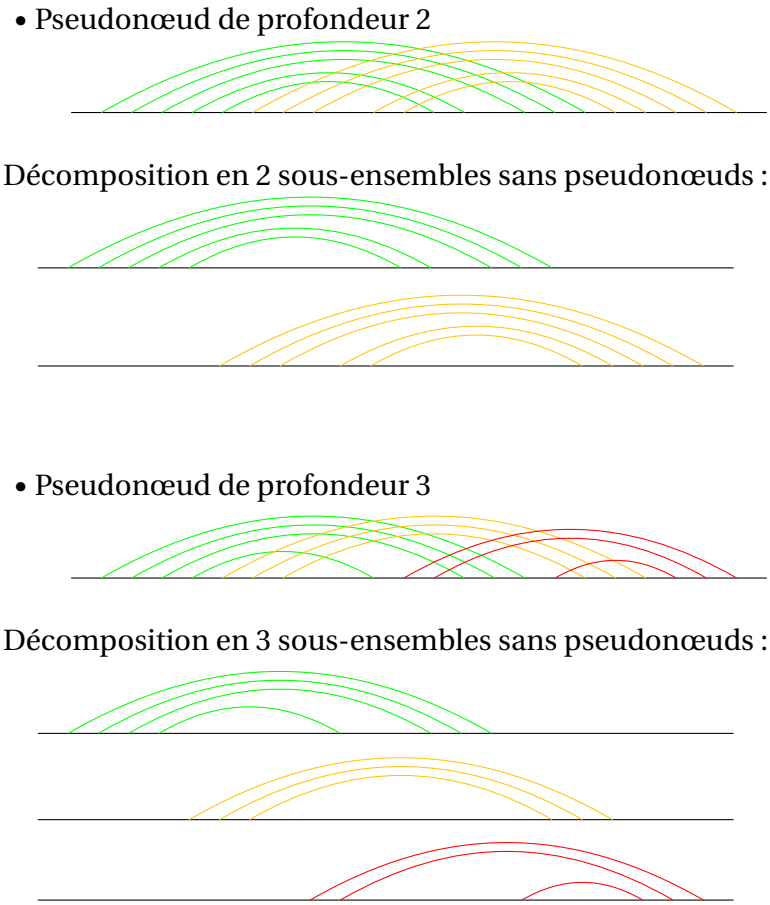

Figure 1.10 : Pseudonœuds de profondeur 2 et 3.

1.1.5 Les complexes d'ARN

Les ARN sont dynamiques et peuvent interagir avec leur environnement dans la cellule. Un ARN peut notamment interagir avec un ou plusieurs ARN pour former des complexes.

Lorsqu'un ARN interagit avec un second ARN, des motifs appariements se forment entre les deux. Les appariements sont appelés appariements d'*hybridation* et l'ensemble de ces appariements forment le *site d'interaction*. La structure secondaire interne à chaque ARN et les sites d'interactions forment ensemble la *structure jointe*. Les appariements d'hybridations peuvent former des hélices, comme lorsqu'un ARN seul se replie, ainsi que des *pseudonœuds externes* (voir figure 1.11). Par opposition, les pseudonœuds au sein d'un ARN seul sont appelés *pseudonœuds internes*.

Nous présentons dans la suite quelques exemples de complexes d'ARN composés de plus de deux ARN.

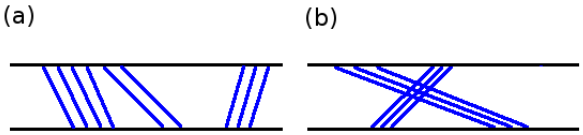

Figure 1.11 : Les motifs de la structure secondaire résultants de l'interaction de deux ARN. Les segments noirs représentent les deux ARN et les segments bleus représentent les appariements. (a) Hélices. (b) Pseudonœud externe.

Le splicéosome Une fois l'ARNm transcrit, il doit subir des modifications post-transcriptionnelles. Une de ces modifications chez les eucaryotes est l'épissage, il s'agit de cliver certaines parties de l'ARN, appelées introns, qui ne codent pas les protéines. Les parties de l'ARN restantes sont les exons. Les exons peuvent être associés en différents groupes, ce qui peut donner plusieurs ARNm et par la suite plusieurs protéines différentes à partir d'un gène identique. Ce type d'épissage est appelé épissage alternatif. Il permet de condenser le stockage de l'information génétique. L'épissage est réalisé par le splicéosome, il s'agit d'un complexe *ribonucléoprotéique*, c'est-à-dire qu'il est composé à la fois de protéines et d'ARN. Le splicéosome est constitué de 5 ARN non codants : U1, U2, U4, U5, U6 et s'associent à plus de 150 protéines. Lors de l'épissage, la structure du splicéosome change et différentes assiociations d'ARN se forment (voir figure 1.12).

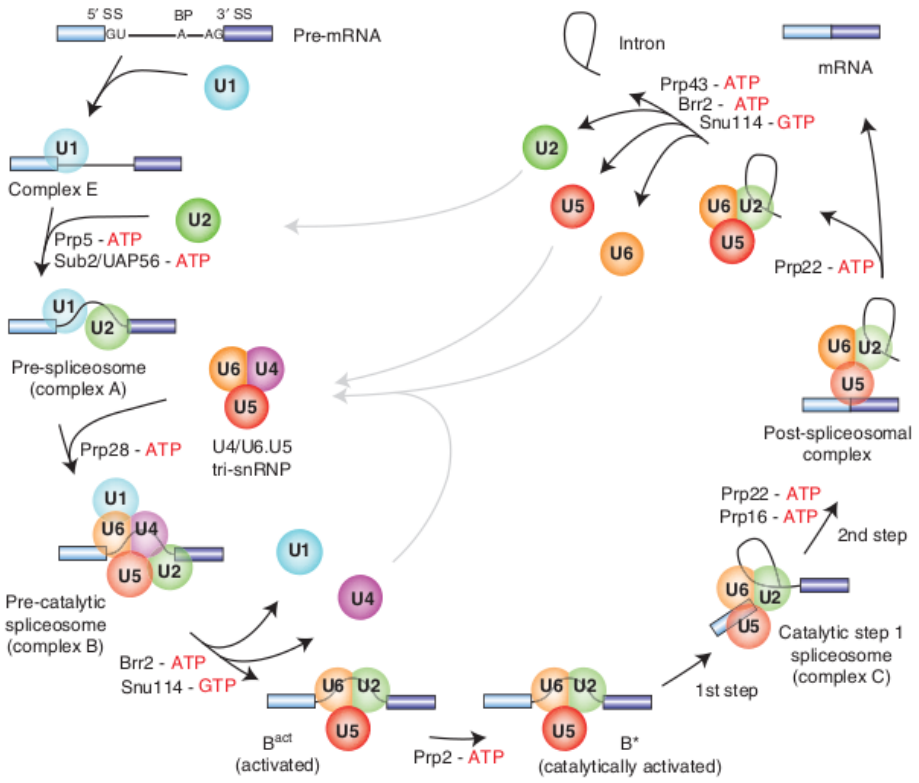

Figure 1.12 : Épissage d'un ARN [234].

Tout d'abord les ARN U1 et U2 se fixent sur l'ARNm et forment ensemble le complexe A : le *pré-splicéosome*. Ensuite les ARN U4, U5 et U6 se rajoutent au complexe A et forment le complexe B : le splicéosome *pré-catalytique*. U1 et U4 sont relachés pour former le complexe B^{act} : le *splicéosome activé*. Le complexe est catalytiquement activé par association avec la protéine Prp2 qui apporte de l'ATP, le complexe est appelé B^{*}. L'excision de l'intron est réalisée par le complexe C et lorsque l'intron est excisé et que l'ARNm est toujours présent, le complexe est appelé complexe *post-splicéosomale*. Différentes structures adoptées par le splicéosome sont présentées en figure 1.13.

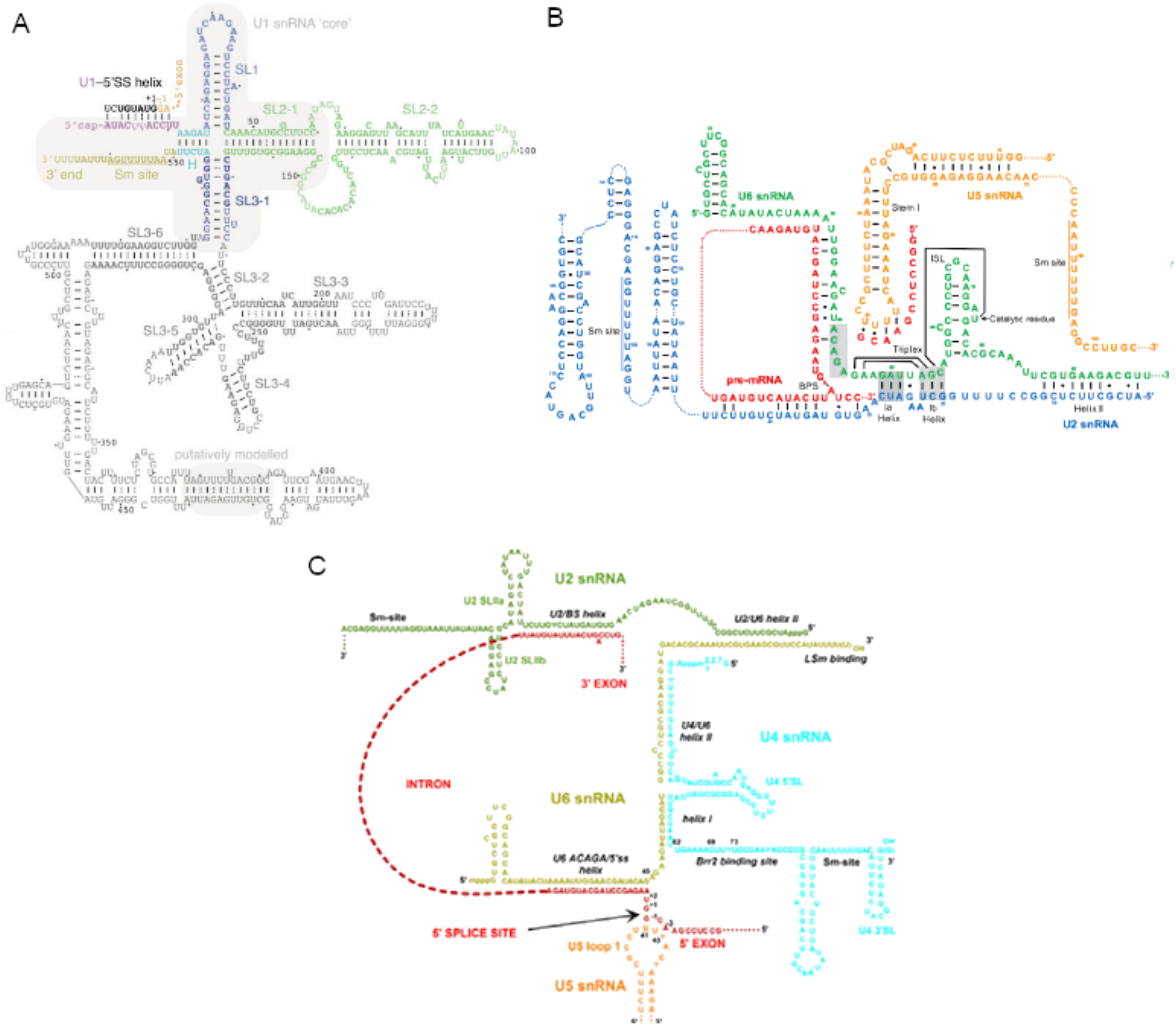

Figure 1.13 : Structures du splicéosome. A) Structure secondaire de l'ARN U1 du complexe A chez la levure [166]. B) Partie de la structure secondaire connue du complexe B chez l'homme [253]. C) Partie de la structure secondaire connue du complexe C chez l'homme [20].

Le ribosome Le ribosome est le complexe responsable de la traduction des ARNm en protéines dans la cellule. Ce complexe est composé d'un grand nombre de protéines et de plusieurs ARN non codants appelés ARN ribosomiques (ARNr). Le ribosome est un ribozyme car sa fonction catalytique est

assurée par un de ses ARN. Le ribosome est présent chez tous les organismes. Sa structure et sa séquence sont très conservées évolutivement et par conséquent ce complexe est utilisé en phylogénie pour déterminer les liens entre les organismes et leur évolution. Lors de la traduction, l'ARNm va être lu par le ribosome. Un triplet de bases azotées de l'ARNm correspond à un acide aminé composant la future protéine. La correspondance entre les triplets de bases et les acides aminés est faite selon le code génétique. Les acides aminés sont apportés dans le site A du ribosome par le biais d'ARN de transfert (ARNt) (voir figure 1.3). Une fois l'ARNt fixé dans le site A du ribosome, le début de la protéine synthétisée (le peptide) est déplacé à la suite du nouvel acide aminé : c'est l'étape de la *transpeptidation*. L'ARNt est ensuite *transloqué* dans le site P du ribosome, d'où le peptide est transpeptidé. Enfin lorsque l'ARNt du site P est libéré de son peptide, il est déplacé dans le site E du ribosome, situé à droite du site P, où il sera ensuite expulsé du ribosome. Le ribosome est composé de deux sous-unités (voir figure 1.14). Chez les eucaryotes, la grande sous-unité est constituée de 49 protéines et de trois ARN : 5S, 5,8S et 28S. La petite sous-unité est constituée de 33 protéines et de l'ARN 18S. La structure secondaire des ARN est illustrée en figure 1.14 A. Dans le ribosome, les ARN interagissent entre eux par des liaisons Watson-Crick mais aussi par des liaisons tertiaires.

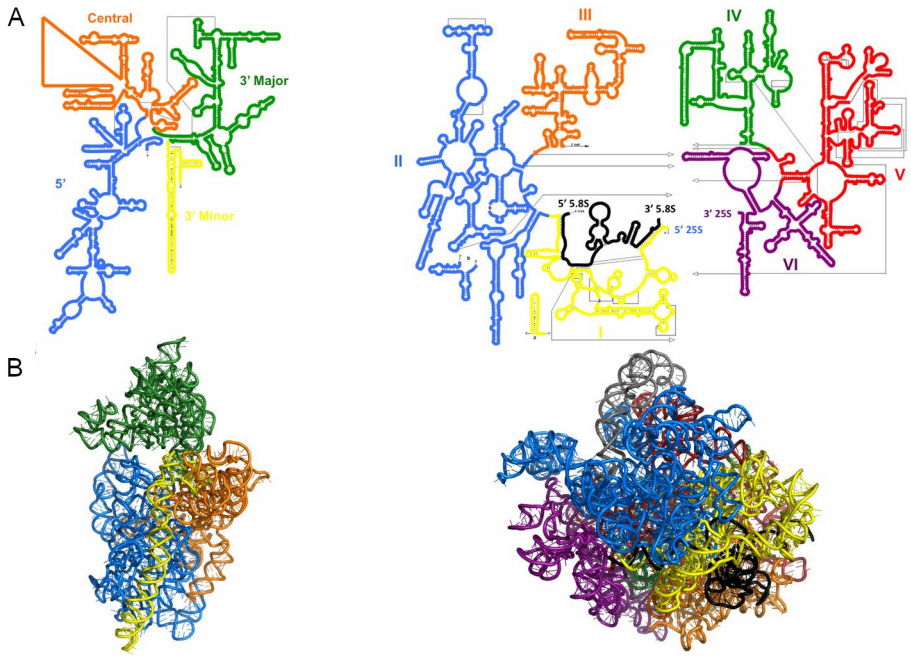

Figure 1.14 : Structures secondaires (A) et structures 3D (B) des ARN des deux sous-unités composant le ribosome de la levure [238].

Le bactériophage φ29 Les bactériophages sont des virus infectant les bactéries [250]. Ils sont constitués d'une capsule (tête) contenant leur génome qui dans la majorité des cas est de l'ADN double brin et d'une queue contractile qui sert d'ancrage pour infecter une cellule. Pour infecter les bactéries, le bactériophage φ29 possède un hexamère d'ARN cyclique pour transporter son génome dans la bactérie hôte. Le complexe est constitué de 6 *prohead RNA* (pRNA) de

120 nucléotides chacun. Les pRNA s'assemblent tout d'abord en dimères avec une interaction de type pseudonœud *kissing hairpin*. Chaque pRNA est en interaction avec deux autres pRNA de manière à former une chaîne et un cycle formant le complexe (figure 1.15). Ce complexe a été une base pour synthétiser des complexes d'ARN appelés tectoARN [57].

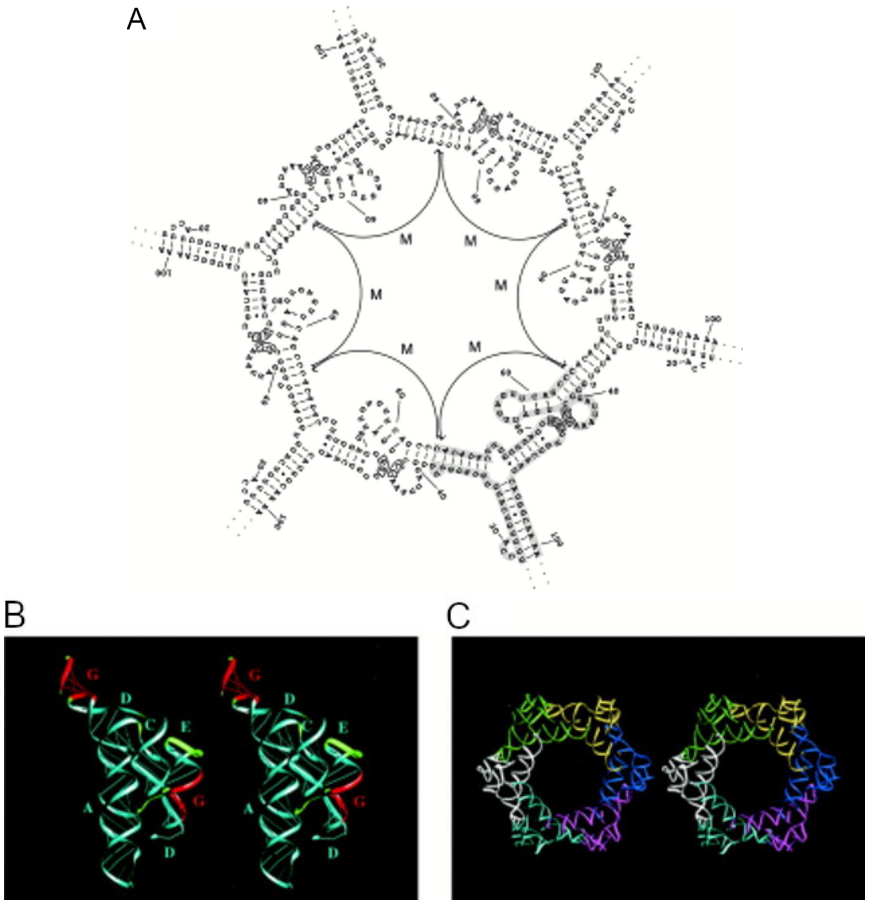

Figure 1.15 : Structure secondaire (a) et modèles en trois dimensions (b, c) de l'hexamère d'ARN servant au transport du génome du bactériophage ϕ29 dans une cellule durant l'infection [250].

Les *tectoRNA* Les tectoRNA [97] sont des complexes d'ARN synthétiques capables de s'assembler de façon autonome à l'échelle du nanomètre ; on parle de nanotechnologie. Grâce à la flexibilité de la structure des ARN, la diversité des structures des tectoRNA est importante. En biologie synthétique, de nombreux ARN appelés riborégulateurs sont créés dans le but de réguler l'expression des gènes avec des mécanismes basés sur les ARN. La figure 1.16 représente la structure d'un tectoRNA formant un carré par l'assemblage de huit ARN [57].

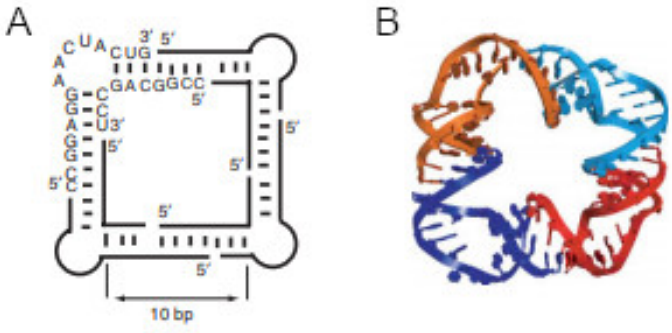

Figure 1.16 : Structure d'un complexe d'ARN synthétique formant un carré [57]. A) Structure secondaire. B) Structure 3D.

1.1.6 Quelques méthodes expérimentales de détermination des structures d'ARN

Détermination de la structure secondaire

Les méthodes expérimentales de détermination de la structure secondaire sont basées soit sur la réaction d'une petite molécule avec l'ARN, soit sur le clivage de l'ARN par une enzyme. Pour la première catégorie de méthodes, le réactif va se fixer sur une face Watson-Crick des nucléotides ou sur le OH du ribose (voir figure 1.17). Lorsqu'un appariement Watson-Crick a lieu entre deux nucléotides, le réactif ne pourra pas se fixer sur ces deux nucléotides. En relevant les positions où le réactif se fixe, on peut alors déterminer quels nucléotides sont appariés ou non. Cependant, nous ne pouvons pas connaître quels sont les couples de nucléotides appariés. Pour chaque nucléotide qui a pu réagir avec le réactif, une donnée est produite. Elle est appelée *réactivité* et donne le taux de réaction à la molécule. Dans les méthodes de la deuxième catégorie, les enzymes vont couper l'ARN au niveau de son squelette entre le ribose et le phosphate [115]. Dans certaines méthodes, deux enzymes sont utilisées, coupant préférentiellement, soit les ARN simples brins, soit doubles brins. Le rapport de coupe entre les deux enzymes permet alors de déterminer la probabilité qu'un nucléotide soit apparié.

Les méthodes interagissant avec le squelette de l'ARN ne permettent pas de déterminer directement si un nucléotide est apparié ou non. En effet, ces nucléotides peuvent également être engagés dans une interaction tertiaire ou non-canonique. De plus, pour les méthodes utilisant un réactif chimique, ce dernier peut ne pas pouvoir accéder à la totalité des nucléotides car lorsque l'ARN est replié, des nucléotides peuvent être dans une cavité et par conséquent ne sont pas exposés au réactif. Ainsi ce type de méthodes fournit une information partielle sur la structure secondaire de l'ARN.

Dans la suite, nous allons décrire plus précisément comment fonctionnent certaines de ces méthodes, à savoir SHAPE, DMS, *inline probing*, PARS, Frag-seq et d'autres méthodes haut-débits.

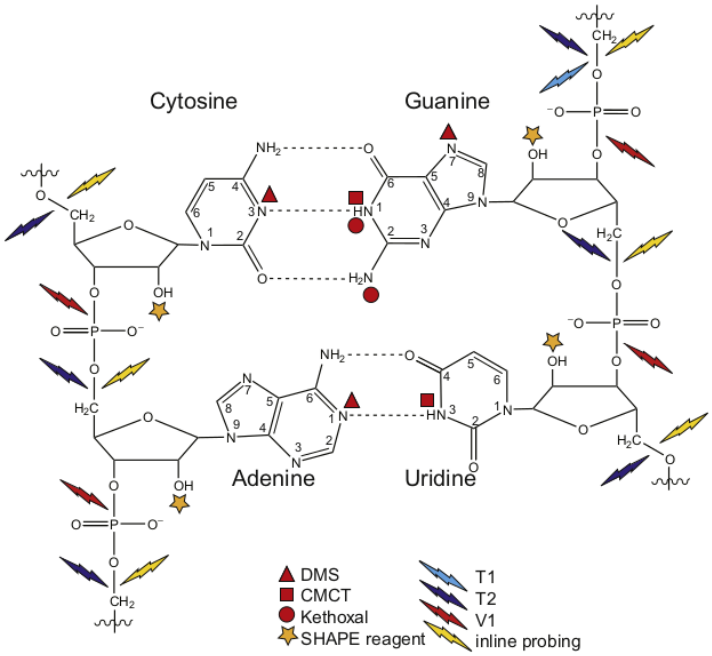

Figure 1.17 : Site d'interaction entre différents réactifs et l'ARN lors de méthodes expérimentales de détermination de la structure secondaire [203]. Le DMS, CMCT, kethoxal et SHAPE sont des réactifs chimiques, tandis que T1, T2 et V1 sont des enzymes. L'*inline probing* est une méthode d'auto-clivage de l'ARN grâce à des cations divalents.

Selective 2-Hydroxyl Acylation analyzed by Primer Extension La méthode *Selective 2-Hydroxyl Acylation analyzed by Primer Extension* (SHAPE) [24] consiste à faire réagir la molécule N-methylisatoic anhydre avec le groupement hydroxyl du carbone 2' de l'ARN. La molécule se fixe lorsque le nucléotide lié à ce groupement est non apparié. La réactivité obtenue par cette méthode est dépendante du contexte structural : suivant le type de boucle et la position dans la boucle d'un nucléotide, la réactivité n'est pas la même [230] ; lorsqu'un nucléotide se trouve dans une hélice, sa réactivité est dépendante de celle des nucléotides voisins [24]. Elle est également influencée par les interactions tertiaires [230]. Les expériences de SHAPE peuvent être réalisées pour un ARN en présence d'un ligand pour déterminer sa structure en interaction.

Dyméthyle sulfate et assimilés Le dyméthyle sulfate (DMS) [43] réagit avec la face Watson-Crick des nucléotides non appariés. Il va méthyler l'azote 1 chez les adénines ou l'azote 3 des cytosines ce qui va permettre de déterminer si ces nucléotides sont impliqués dans des appariements. Le DMS peut également modifier l'azote 7 des guanines, mais cela va donner des informations sur la structure tertiaire. Cette modification peut être détectée par clivage du squelette par l'aniline dye [69]. De la même manière que le DMS, les réactifs kethoxal et *N-Cyclohexyl-N-(2-morpholinoethyl)carbodiimide metho-p-toluenesulfonate* (CMCT) peuvent modifier les nucléotides. Le kethoxal modifie l'amine exocyclique et l'azote 1 des guanines. Le CMCT lui, modifie l'azote 3 de

l'uridine et l'azote 1 des guanines.

Inline probing Le principe de la méthode *inline probing* [178] est l'autoclivage de l'ARN lorsque celui-ci est en présence de cations divalents et d'un pH élevé (entre 8 et 8.5). Le clivage se fait au niveau des phosphates et préférentiellement lorsqu'il n'y a pas d'appariement. Cette méthode permet donc elle aussi de déterminer la probabilité qu'un nucléotide soit non-apparié.

Parallel analysis of RNA structure Le principe de la méthode *parallel analysis of RNA structure* (PARS) [226] est de traiter l'ARN avec deux enzymes séparément :

— L'enzyme RNase V1 coupe préférentiellement les liaisons 3' des ARN doubles brins.
— L'enzyme RNase S1 coupe préférentiellement les liaisons 3' des ARN simples brins.

En faisant le rapport des coupes par les deux enzymes pour chaque nucléotide on peut déterminer si celui-ci était apparié ou non. Cette méthode utilise les technologies de séquençage (PARS-seq) ce qui permet de déterminer des données structurales à l'échelle d'un génome.

Fragmentation sequencing Le *fragmentation sequencing* (FragSeq) [222] est une méthode enzymatique utilisant les technologies de séquençage. Elle permet de déterminer des données structurales à l'échelle génomique. L'enzyme utilisée est la nucléase P1 qui va cliver les ARN préférentiellement non-appariés au niveau de leur squelette.

Méthodes haut-débit Ces méthodes sont les versions hauts-débits des méthodes précédentes, elles ont été développées pour le SHAPE (SHAPE-seq) [131], le DMS (DMS-seq) [61], ou encore pour d'autres réactifs (*multiplexed accessibility probing*, MAP-seq) [196] et pour les méthodes enzymatiques (PARS-seq, Frag-seq) [110, 222]. Elles permettent de mesurer la réactivité de chaque nucléotide d'un grand ensemble d'ARN, notamment à l'échelle transcriptomique. Ces méthodes peuvent notamment être utilisées pour déterminer la structure d'ARN ayant des séquences différentes et qui pourraient interagir entre elles. Le principe est de faire réagir les ARN avec le réactif, puis de les faire transcrire en ADN par une protéine appelée *reverse transcriptase*. La *reverse transcriptase* est stoppée là où le réactif est fixé et va produire des séquences d'ADN codants (ADNc) plus courtes que la longueur totale de l'ARN. Le nombre de séquences d'une longueur donnée va refléter la fréquence d'appariement ou non-appariement d'un nucléotide. Les ADN résultants sont ensuite préparés pour être séquencés : ajout d'adaptateurs et amplification par *polymerase chain reaction* (PCR). Enfin, ils sont séquencés et les *reads* obtenus sont analysés par bioinformatique pour calculer les réactivités de chaque nucléotide d'ARN.

Détermination de la structure tertiaire

Les méthodes de détermination de la structure tertiaire visent à connaître la position des atomes dans l'espace. Nous présentons ici quatre méthodes permettant d'obtenir des informations sur la structure tertiaire : la cristallographie, la microscopie électronique, la résonance magnétique nucléaire et le traitement au radical hydroxyle.

Cristallographie La cristallographie aux rayons X [77], aussi appelée diffraction par rayons X, permet de déterminer la structure en trois dimensions des macromolécules et notamment des macromolécules biologiques comme les protéines et les acides nucléiques. La cristallographie repose sur la formation de cristaux des macromolécules qui dépend de conditions physico-chimiques très variables (pH, tampon, ions, ...). La formation de cristaux de protéines ou d'acides nucléiques peut être difficile à obtenir. Un faisceau de rayons X est ensuite propulsé sur les cristaux d'au moins plusieurs dizaines de microns. Les cristaux vont diffracter les rayons X et des motifs vont se répéter. Ces motifs sont enregistrés et leur analyse va permettre de déterminer les dimensions de la molécule répétée dans l'espace, appelée la *maille du cristal*. L'intensité des motifs, appelée *phase*, est déterminée par différentes méthodes et permet de retrouver par transformation de Fourier la densité électronique des signaux de diffraction. À partir de la densité électronique, les coordonnées dans l'espace des atomes peuvent être retrouvées de manière automatique.

Microscopie électronique La microscopie électronique [41] est basée sur la visualisation de molécules à l'aide de microscopes électroniques qui vont illuminer les molécules avec un faisceau d'électron. La résolution de ces microscopes est très basse ce qui permet d'obtenir la forme générale des protéines et des acides nucléiques. Cependant, la résolution n'est pas encore assez basse pour obtenir la structure 3D totale. Avant de pouvoir visualiser les molécules biologiques, les échantillons doivent être préparés pour les stabiliser, pour réduire leur épaisseur et pour accroître leur contraste de coloration. Cela peut se faire par exemple avec un traitement chimique ou par cryofixation (cryomicroscopie) [206], mais il existe de nombreuses autres méthodes.

Résonance magnétique nucléaire La résonance magnétique nucléaire (RMN) [171] utilise les propriétés magnétiques des atomes des molécules biologiques. Les molécules biologiques en solution sont soumises à des champs magnétiques importants, ce qui va aligner les spins (propriété des atomes liée au moment cinétique) des atomes. À partir des signaux RMN, un spectre va être réalisé à partir duquel des contraintes géométriques atomiques seront déterminées. Ces contraintes permettent de calculer une structure 3D de la molécule d'intérêt.

Radical hydroxyle Le traitement d'ARN au radical hydroxyle [45] permet de déterminer des interactions de la structure tertiaire. Le radical hydroxyle induit un clivage du squelette de l'ARN au niveau de tous les nucléotides. Le fait qu'un

nucléotide soit apparié ou non n'a pas d'influence sur l'efficacité du radical hydroxyle. La fréquence de coupe d'un nucléotide est exprimée par la réactivité au radical hydroxyle. L'analyse de la réactivité permet donc de déterminer des informations de la structure tertiaire de l'ARN, mais pas la structure 3D complète.

1.2 Algorithmique et optimisation combinatoire

1.2.1 La complexité

La complexité d'un algorithme est l'étude des ressources nécessaires à son exécution. On distingue la complexité en temps et en espace mémoire. La complexité exacte est différente pour chaque instance d'un problème. En pratique, on donne une borne de la complexité dans le pire des cas et qui dépend de la taille du problème. On utilise pour cela la notation de Landau $\mathcal{O}(f(n))$ qui représente l'ensemble des fonctions qui sont bornées par $f(n)$ asymptotiquement (une fonction $g(n)$ est bornée asymptotiquement par $f(n)$ s'il existe une constante $M > 0$ et un entier n_0 tels que pour tout $n > n_0$, $|f[n)| \leq M \times |g(n)|$).

Dans la théorie de la complexité, les problèmes sont classés suivant leur difficulté à les résoudre (voir figure 1.18), autrement dit, suivant la complexité des algorithmes pour les résoudre.

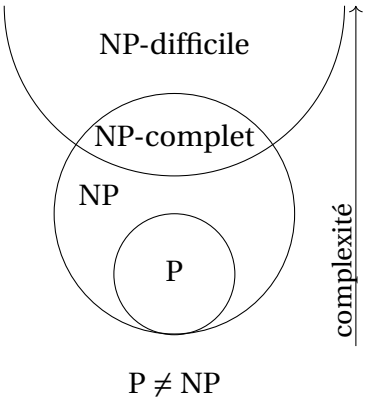

Figure 1.18 : Représentation des classes de problèmes dans la théorie de la complexité. Les problèmes de classe P sont des problèmes polynomiaux. Les problèmes de classe NP sont des problèmes non-déterministes polynomiaux. Un problème est NP-difficile si tous les problèmes NP se ramènent à lui par une réduction polynomiale. Un problème est NP-complet s'il est NP et NP-difficile.

Définition 1.2.1. Les problèmes de classe P sont des problèmes *polynomiaux*, c'est-à-dire qu'il existe des algorithmes en temps polynomial pour les résoudre.

Les problèmes de classe NP sont des problèmes *non-déterministes polynomiaux*, c'est-à-dire qu'il existe un algorithme en temps polynomial permettant de vérifier qu'une solution au problème est valide (mais pas nécessairement de la trouver).

Une *réduction polynomiale* d'un problème A en un problème B se réalise par une paire de fonctions calculables en temps polynomial permettant :

— de transformer une instance de A en une instance de B,

— et de transformer la solution de B en une solution de A.

Un problème de décision est *NP-difficile* si tous les problèmes NP se ramènent à lui par une réduction polynomiale. Par conséquent, un problème NP-difficile est au moins aussi difficile que tous les problèmes de NP.

Un problème de décision est *NP-complet* s'il est NP et NP-difficile.

Un problème d'optimisation combinatoire est *NP-difficile* si le problème de décision associé est NP-complet.

Déterminer si $P = NP$ est un problème majeur en informatique. Il n'existe pas de preuve actuellement, mais la majorité de la communauté scientifique travaillant sur ce problème pense que $P \neq NP$. Si la classe P est effectivement différente de NP, il n'existe pas d'algorithme de complexité en temps polynomial pour résoudre les problèmes NP-difficiles.

Pour trouver des solutions à des problèmes NP-difficiles, il est possible d'utiliser des algorithmes polynomiaux permettant de trouver des solutions proches de la solution optimale d'un problème. Ces algorithmes sont dits *approchés*.

Définition 1.2.2. Pour un problème de maximisation (respectivement minimisation) un algorithme est dit *α-approché*, si pour toute instance du problème, il trouve une solution x telle que son coût est supérieur ou égal (respectivement inférieur ou égal) à α multiplié par le coût d'une solution optimale pour l'instance.

1.2.2 Les graphes

Définitions et propriétés des graphes

Nous définissons ici les graphes en général et certaines de leur propriétés. La définition d'un graphe quelconque est la suivante :

Définition 1.2.3. Un *graphe* $G(V, E)$ est défini par un ensemble de sommets V et un ensemble d'arêtes E reliant des sommets. Une arête est une paire de sommets $\{u, v\}$ (une paire $\{u, v\}$ est identique à la paire $\{v, u\}$).

Un graphe peut également être définit à partir d'un autre graphe, c'est le cas des graphes *induits*, des graphes *complémentaires* ou encore des graphes *isomorphes*.

Définition 1.2.4. Le graphe *induit* $G_i(V_i, E_i)$ par un sous-ensemble de sommet $V_i \subseteq V$, par rapport à un graphe $G(V, E)$, est tel qu'une arête $\{u, v\}$, avec u et $v \in V_i$, existe dans E_i si et seulement si elle existe dans E.

Définition 1.2.5. Le graphe *complémentaire* G_c (ou *inversé*) d'un graphe G possède les même sommets que le graphe G et une arête existe dans G_c seulement si elle n'existe pas dans G.

Définition 1.2.6. Un *isomorphisme* f entre les graphes G et H est une bijection entre les sommets de G et ceux de H, telle qu'une paire de sommets $\{u, v\}$ de G est une arête de G si et seulement si $\{f(u), f(v)\}$ est une arête de H.

Nous définissons maintenant quelques propriétés des graphes importantes pour la compréhension de la suite du manuscrit.

Définition 1.2.7. Un graphe est dit *complet* si toutes les arêtes possibles existent. Une *clique* d'un graphe est un sous-graphe complet.

Définition 1.2.8. Un graphe *connexe* est un graphe dans lequel on peut relier, directement ou non, n'importe quel sommet à n'importe quel autre sommet du graphe par une chaine d'arêtes.

Définition 1.2.9. Une *coloration* d'un graphe est une manière de colorer les sommets d'un graphe de façon à ce que deux sommets reliés par une arête n'aient pas la même couleur. Le *nombre chromatique* d'un graphe est le nombre minimum de couleurs pouvant être utilisées dans une coloration d'un graphe.

Classes particulières de graphes

Nous définissons ici des classes de graphes auxquelles nous nous sommes intéressées lors de cette thèse.

Définition 1.2.10. Un *arbre enraciné* est un graphe acyclique, connexe, possédant une racine unique et tel que tous les sommets ont un parent unique, sauf la racine. Les sommets ne possédant pas d'enfants sont appelés les feuilles. Un arbre enraciné est *binaire* si chaque sommet, sauf les feuilles, possède exactement deux enfants.

Définition 1.2.11. Un *graphe d'intervalles* est le graphe d'intersection d'un ensemble d'intervalles de la droite réelle. Pour tout intervalle I, il existe un sommet v_I dans le graphe, et il existe une arête entre deux sommets v_I et $v_{I'}$, si et seulement si, l'intersection entre les deux intervalles I et I' est non nulle.

Les structures secondaires d'ARN peuvent être assimilées à des graphes de cercle [102]. Ces graphes peuvent être utilisés pour calculer la profondeur des pseudonœuds [42].

Définition 1.2.12. Un *graphe de cercle* est similaire à un graphe d'intervalles en enlevant les arêtes correspondantes aux intervalles inclus les uns dans les autres. C'est un graphe non-orienté dans lequel les sommets peuvent être associés à des cordes d'un cercle, tels que deux sommets sont adjacents si et seulement si les cordes correspondantes se croisent.

La figure 1.19 montre comment est construit un graphe de cercle pour contrôler la profondeur des pseudonœuds de deux ARN, A et B, pour lesquels nous avons trois hélices. Trois hélices sont représentées par des cordes. Nous pouvons voir que l'hélice $i3 - j3$ ne forme pas de pseudonœud avec l'hélice $i2 - j2$: cela a pour effet que la corde $[i3, j3]$ ne croise pas la corde $[i2, j2]$.

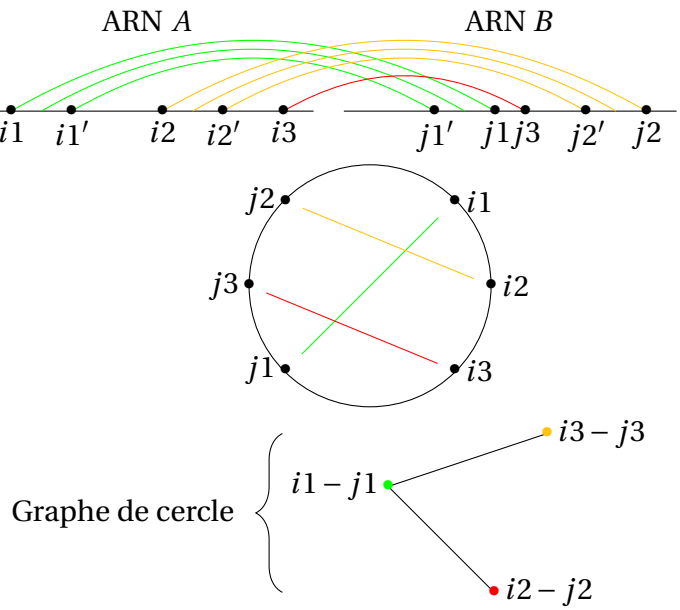

Figure 1.19 : Illustration des graphes de cercle sur deux ARN formant une interaction.

Problèmes de graphes classiques

Détermination de la clique maximum Le problème de la clique maximum [28] consiste à trouver la clique de plus grande taille d'un graphe. Dans la variante pondérée de ce problème, chaque sommet a un poids et on cherche la clique de poids maximum, le poids d'une clique étant égal à la somme des poids des sommets la composant. Ce problème est NP-difficile à résoudre et c'est un problème difficile à approximer. Un algorithme approché cherchant la clique maximum est dit α-approché s'il trouve des cliques de tailles x, telles que $x \geq \alpha x^*$, où x^* est la taille de la clique maximum de ce graphe. Il n'existe pas d'algorithmes polynomiaux permettant de trouver de meilleures solutions que $(n^{1-\epsilon})$-approchées, où n est le nombre de sommets et pour tout $\epsilon > 0$ [89].

Détermination du nombre chromatique La détermination du nombre chromatique [193] consiste à trouver le nombre minimal de couleurs pour colorer un graphe tel que deux sommets reliés par une arête n'aient pas la même couleur. Ce problème est NP-difficile pour des graphes quelconques, comme les graphes de cercle par exemple [42]. Pour certaines classes de graphes, il existe des algorithmes exacts de complexité en temps polynomiale ; par exemple, pour les graphes d'intervalles, il existe un algorithme linéaire [35].

1.2.3 Optimisation combinatoire

Un problème d'optimisation se définit comme la recherche d'un minimum ou d'un maximum d'une fonction dont les variables peuvent être contraintes. La fonction est appelée fonction *objectif* et elle est définie sur des *variables de décision*. Les contraintes d'un problème d'optimisation peuvent être fortes ou faibles.

Définition 1.2.13. Une contrainte *forte* d'un problème d'optimisation doit être absolument respectée dans les solutions de ce problème. Une contrainte *faible*, en revanche, n'est pas forcément respectée dans les solutions au problème.

Nous présentons dans un premier temps la programmation mathématique linéaire qui est une manière d'exprimer et de résoudre des problèmes d'optimisation. Puis, nous présentons l'optimisation multi-objectif qui permet de résoudre des problèmes multi-critères.

La programmation mathématique linéaire

La *programmation mathématique* permet de modéliser et de résoudre des problèmes d'optimisation. En programmation mathématique, les contraintes se présentent sous la forme d'équations ou d'inéquations. L'ensemble des contraintes et la fonction objectif forment un système d'équations, appelé *programme mathématique*, dont nous cherchons la solution qui optimise la fonction objectif. Si la fonction objectif et les contraintes sont linéaires, alors le programme mathématique est dit *linéaire* [235]. Par simplicité, nous parlerons par la suite de *programmation linéaire*. Lorsque les variables de décisions prennent des valeurs entières, le programme mathématique est dit *en nombres entiers*. S'il existe à la fois des variables à valeurs entières et des variables à valeurs réelles, le programme mathématique est dit *mixte*.

Un problème linéaire en nombres entiers (PLNE) quelconque peut donc se définir de la manière suivante :

$$\min f(x)$$

tel que :

$$g_k(x) \leq 0 \qquad k = 1, \ldots, m$$

$$x \in \mathbb{Z}^n$$

où $f(x) : R^n \mapsto R$ est la fonction objectif linéaire à minimiser et où les contraintes sont décrites comme des fonctions linéaires g_k de x.

Grâce à sa maniabilité, la programmation linéaire permet de décrire de nombreux problèmes comme la planification, le contrôle de chaînes de production ou des réseaux de distribution. C'est pour cela que la programmation linéaire est utilisée dans divers domaines : l'énergie, les transports, la communication, et maintenant en bioinformatique.

Pour résoudre un problème linéaire, il existe des algorithmes exacts tels que l'algorithme du simplex [47] ou l'algorithme de points intérieurs [145]. Ces algorithmes sont implémentés dans des solveurs comme GLPK [1], GUROBI [85], ou CPLEX [96].

La complexité en temps de la résolution d'un problème linéaire dépend du type des variables de décision, de leur nombre et du nombre de contraintes. En effet, lorsque les variables sont entières, la complexité est exponentielle tandis que lorsque les variables sont réelles, le problème peut être résolu en temps polynomial. Ainsi, il existe des méthodes approchées et des heuristiques pour résoudre des problèmes entiers. Parmi les méthodes approchées, la relaxation lagrangienne [15] ou la relaxation continue sont utilisées et permettent

parfois d'obtenir des solutions proches de l'optimale avec une garantie de performance.

L'optimisation multi-objectif

En optimisation multi-objectif, on cherche à résoudre des problèmes d'optimisation avec plusieurs critères. Lorsque l'on résout un problème multi-critère, on obtient le plus souvent une multitude de solutions du fait que les critères peuvent être contradictoires. Si une solution permet d'optimiser tous les objectifs simultanément alors elle est optimale, mais en général, plusieurs solutions vont former une zone de compromis. Ces solutions vont optimiser certains critères et en dégrader d'autres. L'optimalité de ces solutions est définie grâce à la relation de *dominance* entre les solutions : les solutions non-dominées sont les solutions optimales. Les solutions non-dominées forment une zone de compromis entre tous les critères, c'est l'*ensemble de Pareto*.

Définition 1.2.14. Une solution x *domine* une solution y, noté $x \succ y$, si
- x est au moins aussi bonne que y dans tous les objectifs, et,
- x est strictement meilleure que y dans au moins un objectif.

Définition 1.2.15. L'*ensemble de Pareto* est l'ensemble des solutions non-dominées.

Définition 1.2.16. Le *front de Pareto*, aussi appelé *courbe de Pareto*, est l'ensemble des valeurs objectifs des solutions non-dominées.

Ces notions sont illustrées à la figure 1.20 pour un problème bi-objectif.

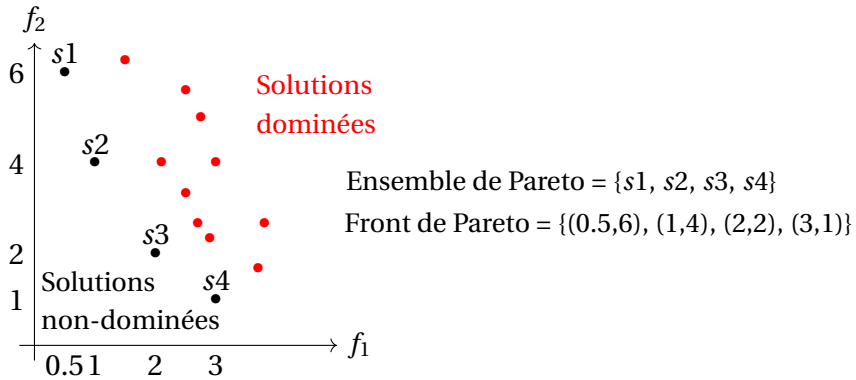

Figure 1.20 : Illustration de l'ensemble et du front de Pareto pour un problème bi-objectif où les deux objectifs f_1 et f_2 doivent être minimisés.

Une courbe de Pareto ϵ-approchée est un ensemble de solutions qui dominent de manière approchée toutes les autres solutions, c'est-à-dire, l'ensemble contient, pour chaque solution de la courbe de Pareto exacte, une solution qui est au moins aussi bonne avec un facteur $1+\epsilon$ dans tous les objectifs par rapport à celle-ci.

Définition 1.2.17. Une *courbe de Pareto ϵ-approchée* est telle que (pour des critères à minimiser) : pour toute solution optimale x^* d'une courbe de Pareto exacte, il existe au moins une solution x de la courbe de Pareto ϵ-approchée telle que pour tout critère j, $f_j(x) \leq (1+\epsilon) f_j(x^*)$ (respectivement $\geq$ pour des critères à maximiser).

Parmi les solutions optimales de la courbe de Pareto, on distingue deux types de solutions : les solutions *supportées* et *non-supportées*. Ces notions sont illustrées en figure 1.21.

Définition 1.2.18. Une solution optimale est *supportée* si elle se situe sur l'enveloppe convexe de l'ensemble des solutions Pareto. À l'inverse, une solution optimale est *non supportée* si elle ne se situe pas sur l'enveloppe convexe de l'ensemble des solutions.

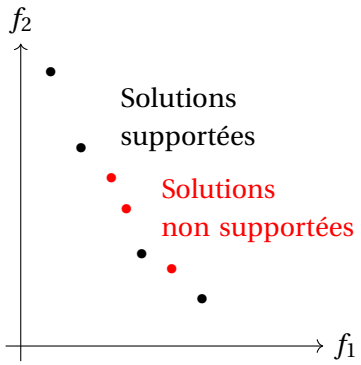

Figure 1.21 : Illustration des notions de solutions supportées et non supportées pour un problème bi-objectif où les deux objectifs f_1 et f_2 doivent être minimisés.

Définition 1.2.19. Dans un problème multi-objectif, le *point idéal* est un point imaginaire ayant pour coordonnées les meilleures valeurs de chacun des critères. Le *point nadir* est un point imaginaire ayant pour coordonnées les pires valeurs de chacun des critères au sein du front de Pareto.

Définition 1.2.20. Le k^e *ensemble de Pareto* est l'ensemble des solutions non-dominées d'un problème multi-objectif lorsque les solutions des $(k-1)$ meilleurs ensembles de Pareto sont ignorées (voir Figure 1.22).

1.3 Bioinformatique des ARN

1.3.1 Approches bioinformatiques pour la prédiction de structures secondaires des ARN

La prédiction de structures secondaires des ARN repose principalement sur des modèles basés sur la thermodynamique. Les premières méthodes de détermination de la structure visent à trouver la structure ayant l'énergie libre minimum : c'est le modèle MFE (*Minimum Free Energy*). Plus tard, cette méthode

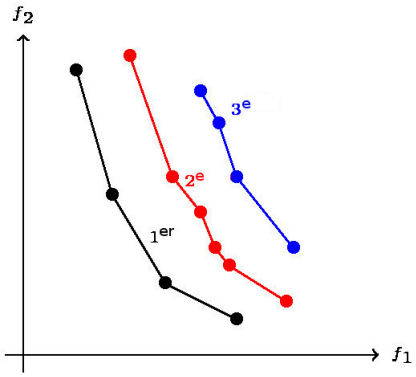

Figure 1.22 : Illustration des 3 ensembles de Pareto pour un problème bi-objectif où les deux objectifs f_1 et f_2 doivent être minimisés.

a été améliorée et grâce au calcul de la fonction de partition, les probabilités d'appariements entre nucléotides ont pu être estimées. Ces probabilités sont utilisées par le modèle MEA (*Maximisation of Expected Accuracy*).

D'autres méthodes sont basées sur l'approche comparative. L'approche comparative consiste à tirer parti de informations évolutives de séquences d'ARN homologues à un ARN d'intérêt. Les séquences homologues sont des séquences du même ARN chez des espèces proches au niveau évolutif. Cette méthode permet de déterminer la structure d'un ARN d'intérêt ou la structure consensus de l'ensemble des séquences homologues. Les séquences homologues doivent être alignées pour déterminer où se situent les domaines (sous-séquences) conservés et les différences. La séquence consensus d'un ensemble de séquences homologues se définie comme la structure moyenne de ces séquences.

Durant cette thèse, nous nous sommes intéressés à la prédiction de la structure d'un ARN à partir de sa séquence nucléotidique. En d'autres termes, nous avons écarté ici l'approche comparative. La raison est que pour les prédictions de novo, souvent peu de séquences homologues sont disponibles.

1.3.2 L'approche thermodynamique

La structure secondaire d'un ARN est formée par ses appariements. Comme indiqué ci-dessus, la prédiction de ces appariements par informatique se base la plupart du temps sur leur thermodynamique.

En général, les molécules s'agencent de manière à trouver un équilibre thermodynamique stable, c'est-à-dire qu'à une pression et une température constante, les molécules n'ont plus tendance à réaliser de réactions. Une molécule atteint un équilibre thermodynamique lorsque son *énergie libre* est très basse.

Définition 1.3.1. L'*énergie libre de Gibbs G* d'un système est le potentiel thermodynamique qui peut être utilisé par un autre système pour effectuer des transformations réversibles : $G = H - TS$ où :

- H est l'enthalpie. L'*enthalpie H* est le potentiel thermodynamique d'un système, c'est à dire la quantité de chaleur qui peut être libérée lors d'une transformation. Elle s'exprime, pour une réaction chimique, en kilojoule par mole.
- S est l'entropie. L'*entropie S* mesure le degré de dispersion d'un système, c'est une mesure de désordre d'un système. Elle s'exprime, pour une réaction chimique, en joule par mole par kelvin.
- T est la température.

Elle s'exprime, pour une réaction chimique, en kilojoule par mole.

Pour un ARN, nous pouvons calculer l'énergie libre d'une certaine structure. Cette énergie peut être approximée en sommant les contributions énergétiques de petits éléments structuraux. Ces contributions énergétiques peuvent être déterminées expérimentalement. Les éléments structuraux sont les boucles et les hélices et leur contribution énergétique dépend de la nature des nucléotides les constituant et de leurs agencements. Les hélices apportent de la stabilité à la structure et ont une énergie négative contrairement aux boucles.

1.3.3 Le modèle MFE (*Minimum Free Energy*)

Ce modèle [157] a été le premier développé pour la prédiction de la structure secondaire d'ARN. Dans la variante la plus simple, l'objectif est de maximiser le nombre d'appariements qui stabilisent la structure d'un ARN, ce qui revient à minimiser l'énergie. L'énergie d'une séquence de nucléotides allant de i à j peut s'exprimer de manière récursive par la minimisation de l'énergie de plusieurs sous-séquences, selon l'équation 1.1 (voir figure 1.23), et peut être calculée par programmation dynamique.

$$E_{i,j} = \min\{E_{i+1,j}; \min_{k:i+m<k\leq j} E_{i+1,k-1} + E_{k+1,j} + \beta_{i,k}\} \tag{1.1}$$

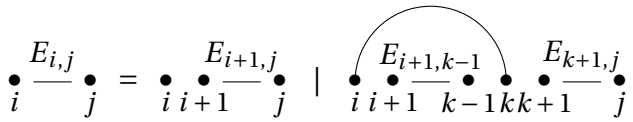

Figure 1.23 : Calcul de l'énergie libre d'une séquence $i\,j$ selon l'algorithme de Nussinov [157].

L'appariement G-C est plus stable que tous les autres appariements car il y a trois liaisons hydrogènes entre les bases. En prédiction bioinformatique, la plupart des méthodes cherchent à prédire les appariements d'une séquence qui minimise l'énergie libre en se basant sur l'énergie des éléments structuraux.

Les paramètres énergétiques des appariements ont été déterminés par Salser en 1978 [192] puis les paramètres énergétiques des hélices et des boucles ont été déterminés dans les années 80 [219] (voir tableaux 1.1 et 1.2). Ces paramètres ont été raffinés d'année en année. Des paramètres ont été ajoutés

suivant le type des motifs et leur composition en nucléotides [198], notamment pour les pseudonœuds [12], les duplexes d'ARN [243] ou encore des contraintes de nature chimique [140]. Les paramètres rassemblés par l'équipe de Turner se trouvent sur la base de données *Nearest Neighbor Database* [218]. Cette base de données donne deux versions de paramètres (1999 et 2004) ainsi que les méthodes de calcul de l'énergie libre des structures d'ARN.

	A/U	C/G	G/C	U/A	G/U	U/G
A/U	-0.9	-1.8	-2.3	-1.1	-1.1	-0.8
C/G	-1.7	-2.9	-3.4	-2.3	-2.1	-1.4
G/C	-2.1	-2.0	-2.9	-1.8	-1.9	-1.2
U/A	-0.9	-1.7	-2.1	-0.9	-1.0	-0.5
G/U	-0.5	-1.2	-1.4	-0.8	-0.4	-0.2
U/G	-1.0	-1.9	-2.1	-1.1	-1.5	-0.4

Tableau 1.1 : Paramètres énergétiques d'empilement d'appariements [219].

Nombre de bases	1	5	10	20	30
Boucle interne	-5.3	6.6	7.0	7.4	-
Renflement	3.9	4.8	5.5	6.3	6.7
Épingle à cheveux	-4.4	5.3	6.1	6.5	-

Tableau 1.2 : Paramètres énergétiques de boucles [198].

Au fil de l'amélioration des paramètres énergétiques, de nouveaux algorithmes de programmation dynamique ont été développés pour calculer la structure d'énergie libre minimale d'une séquence. Ces algorithmes [259, 258] prennent en compte les nouveaux paramètres énergétiques mais apportent également des améliorations techniques : des structures sous-optimales peuvent être retournées [256] et la complexité diminue ($\mathcal{O}(n^2)$ en temps et $\mathcal{O}(n^3)$ en espace).

1.3.4 Le modèle MEA (*Maximum Expected Accuracy*)

Dans les années 90, un nouvel algorithme est développé permettant de calculer la fonction de partition d'une séquence [144]. La fonction de partition (Équation 1.2), notée Z, est une quantité qui englobe les propriétés statistiques d'une séquence à l'équilibre thermodynamique :

$$Z = \sum_{s \in \omega} e^{\frac{-E(P)}{RT}}, \tag{1.2}$$

où w est l'ensemble des structures possibles d'une séquence, appelé *ensemble de Boltzmann*, R est la constante des gaz parfaits, T la température, s une structure et $E(s)$ son énergie.

La probabilité d'une structure s, notée p_s, est définie de la manière suivante :

$$p_s = \frac{\exp -E(s)/RT}{Z}, \tag{1.3}$$

Les probabilités d'appariements entre deux nucléotides i et j, notées $p_{i,j}$, et avec $i < j$, sont définies de la manière suivante :

$$p_{i,j} = \sum_{s \in \omega \text{ t. q. } (i,j) \in s} p_s, \tag{1.4}$$

La probabilité qu'un nucléotide soit non-apparié, notée p_i, peut être calculée par l'équation :

$$p_i = 1 - \sum_{k<i} p_{k,i} - \sum_{i<j} p_{i,j}. \tag{1.5}$$

À partir de ces probabilités, on peut calculer la structure qui maximise la précision attendue (*MEA, Maximum Expected Accuracy*) [65]. La précision maximum attendue d'une séquence allant de i à j inclus, est définie à partir des probabilités d'appariement, notées P_{bp}, et à partir des probabilités de non appariement, notées P_{ss} :

$$\text{MEA}_{i,j} = \max \sum_{(i,j)\in BP} \gamma \times 2P_{bp}(i,j) + \sum_{k\in SS} P_{ss}(k), \tag{1.6}$$

où $P_{bp}(i,j)$ est la probabilité d'appariement (bp pour *base-pair*) d'un nucléotide i avec un nucléotide j, $P_{ss}(k)$ est la probabilité que le nucléotide k soit non-apparié (ss pour *single-stranded*), BP est l'ensemble des appariements et SS est l'ensemble des nucléotides donnés pour une structure. La constante γ sert à pondérer les probabilités d'appariement.

Le calcul de la fonction de partition peut également servir pour échantillonner des structures secondaires. L'échantillonnage de structures secondaires d'ARN est une technique permettant de générer un grand nombre de structures [190, 68]. Il se base sur des grammaires hors-contextes probabilistes et est basé sur le calcul de la fonction de partition. Une grammaire hors-contexte probabiliste permet de modéliser les structures secondaires comme des phrases, dont les mots sont écrits par des règles et dont l'enchaînement peut être analysé par des arbres. Chaque mot a une certaine probabilité d'apparaître dans une phrase. À partir de cette distribution de probabilité, on peut calculer les probabilités des différentes phrases et également la phrase la plus probable. Cela permet de calculer les structures secondaires.

1.4 Définition des métriques

Pour évaluer la qualité d'une structure prédite, les statistiques couramment utilisées sont la sensitivité, la valeur prédictive positive (PPV pour *positive predictive value*), le F_1-score et le coefficient de corrélation de Matthews (MCC pour *matthews correlation coefficient*). La sensitivité mesure la capacité à trouver des appariements vrais positifs, tandis que le PPV mesure la capacité à trouver des appariements faux positifs. Le F_1-score est la moyenne harmonique entre la sensitivité et le PPV. Leur valeurs varient entre 0 % (prédiction complètement fausse) et 100 % (prédiction parfaite). L'avantage du MCC par rapport au F_1-score est qu'il permet de mesurer la qualité d'une prédiction même si les classes (positif et négatif) sont de tailles très différentes. Sa valeur varie entre -1 (prédiction complètement fausse) et 1 (prédiction parfaite). La précision mesure la distance d'une solution par rapport à une référence. Elle varie entre 0 (prédiction complètement fausse) et 1 (prédiction parfaite). Ces métriques permettent toutes de mesurer la qualité d'une solution par rapport à une structure de référence. Ces mesures sont calculées de la manière suivante :

$$\text{Sensitivité} = \frac{TP}{TP+FN}, \qquad \text{PPV} = \frac{TP}{TP+FP},$$

$$\text{F}_1\text{-score} = 2 \times \frac{\text{Sensitivité} \times \text{PPV}}{\text{Sensitivité} + \text{PPV}},$$

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}},$$

$$\text{Précision} = \frac{TP + TN}{TP + TN + FP + FN}.$$

où TP est le nombre d'appariements vrais positifs, FN est le nombre d'appariements faux négatifs, FP est le nombre d'appariements faux positifs et TN le nombre d'appariements vrais négatifs. Un appariement prédit est vrai positif si c'est un appariement existant dans la structure de référence. Les appariements faux négatifs sont les appariements de la structure de référence qui ne sont pas prédits. Un appariement prédit est faux positif s'il n'existe pas dans la structure de référence. Les appariements vrais négatifs sont l'ensemble de tous les appariements possibles moins les TP, FN et FP, c'est-à-dire $TN = \frac{n(n-1)}{2} - TP - FN - FP$ où n est la longueur de l'ARN.

Dans la suite de cette thèse, nous étudions des algorithmes permettant de retourner plusieurs solutions selon un rang. Pour déterminer la qualité d'un ensemble de n solutions selon une structure de référence, nous proposons d'adapter ces mesures de la manière suivante :

$$M = \frac{\sum_{i=1}^{n} M(s_i) \times (n - i + 1)}{n}, \tag{1.7}$$

où M est une mesure, par exemple le F_1-score d'un ensemble de solutions. Elle est calculée en fonction des mesures $M(s_i)$ qui sont les mesures des solutions s_i, pondérées par le rang i de la solution. Plus le rang de la solution est petit, plus la solution est importante, car ce critère est alors optimisé.

2 Prédiction de structures secondaires d'ARN avec pseudonœuds

La prédiction bioinformatique de la structure secondaire d'un ARN donné est un domaine riche et constamment en évolution. De nombreux outils ont été développés pour permettre de prédire des structures toujours plus précisément et plus rapidement. Pour prédire des structures plus précisément, la prise en compte des pseudonœuds est importante. Les structures sous-optimales sont également importantes car les ARN peuvent avoir plusieurs structures stables, notamment à cause de leur environnement qui va les faire changer de conformation. De plus, les structures de références ne correspondent pas toujours aux structures optimales des modèles. Parmi les outils existants, peu sont capables de proposer à la fois des structures sous-optimales et des structures ayant des pseudonœuds. De plus, ces outils sont pour la plupart basés sur un seul modèle de prédiction, les principaux modèles ou approches de prédiction étant : le modèle thermodynamique avec minimisation de l'énergie libre (MFE), le modèle thermodynamique avec maximisation de la précision attendue (MEA), et l'approche comparative.

Afin de proposer un outil capable de prédire plus précisément des structures optimales et sous-optimales avec pseudonœuds, nous avons développé une méthode combinant deux modèles de prédiction. Combiner deux modèles permet de générer des structures secondaires spécifiques à chaque modèle et d'en générer des nouvelles répondant à ces deux modèles.

Nous avons choisi comme modèles le MFE et le MEA car nous nous intéressons à la prédiction de structure à partir d'une seule séquence. Le modèle MEA est en général plus performant que le modèle MFE [134, 88], mais pas toujours, comme nous le montrons dans nos résultats en section 2.3.2. Ces deux modèles ne prédisent pas les mêmes structures (voir également section 2.3.2), d'où l'intérêt de les combiner afin d'extraire les meilleures prédictions de chacun d'eux, et donc de tirer profit de chacun des deux modèles. Pour combiner ces deux modèles, nous avons utilisé la programmation linéaire bi-objectif, où chaque modèle est pris en compte à l'aide d'une fonction objectif. En effet, chacun de ces modèles de prédiction donne lieu à un problème d'optimisation pouvant être exprimé par un programme linéaire. Par exemple, la fonction objectif du

modèle MFE que nous avons utilisé correspond à la somme des énergies d'appariements. Les contraintes dans ces programmes linéaires visent à interdire, par exemple, des appariements entre plus de deux nucléotides. Dans ce chapitre, nous présentons notre programme linéaire bi-objectif combinant le MFE et le MEA.

Des méthodes existent pour générer les solutions optimales exactes d'un programme linéaire bi-objectif, mais il n'en existe aucune pour générer des solutions sous-optimales exactes. Or comme nous l'avons mentionné précédemment, la biologie nous pousse à considérer les solutions sous-optimales. Pour répondre à ce besoin, nous avons développé un algorithme original de résolution de problèmes bi-objectifs, permettant de déterminer de manière exacte l'ensemble de Pareto ainsi que des ensembles de Pareto sous-optimaux, c'est-à-dire les k meilleurs ensembles de Pareto, avec $k \geq 1$ une constante fixée par l'utilisateur.

À partir de notre programme linéaire bi-objectif et de notre algorithme pour générer les k meilleurs ensembles de Pareto, nous avons développé un outil de prédiction de structures secondaires d'ARN à partir d'une séquence. Nous avons appelé cet outil BiokoP, pour *Bi-objective programming pseudoknot prediction*. Contrairement à tous les autres outils de l'état de l'art, BiokoP, grâce à la programmation linéaire, peut prédire tous les types de pseudonœuds et générer des structures sous-optimales tout en combinant les deux modèles MFE et MEA.

La première section de ce chapitre est consacrée à l'état de l'art des outils de prédiction de structures secondaires des ARN. La deuxième section explique le choix des modèles utilisés pour le problème linéaire bi-objectif modélisant le problème de prédiction de structures secondaires, ainsi que notre programme linéaire bi-objectif et notre algorithme pour déterminer les k meilleurs ensembles de Pareto. Une analyse de cet algorithme sera décrite au chapitre 5. Dans la troisième section est présenté l'évaluation de notre outil BiokoP sur un ensemble d'ARN avec pseudonœuds. Enfin, la dernière section est dédiée à la discussion.

2.1 État de l'art

Il existe de nombreuses méthodes et outils prédisant des structures secondaires d'ARN. Nous nous intéressons ici à des méthodes et outils basés sur la thermodynamique, car nous voulons prédire des structures à partir d'une seule séquence. L'approche comparative est donc écartée. Les méthodes prenant en compte des données biologiques structurales telles que le SHAPE feront l'objet d'un état de l'art au chapitre 4.

Nous présentons les méthodes suivant leur capacité à prédire des pseudonœuds, leur capacité à prédire des structures sous-optimales et le modèle de prédiction utilisé.

Les pseudonœuds sont des motifs parfois considérés comme faisant partie de la structure tertiaire. Les premiers algorithmes n'en tenaient souvent pas compte car les inclure dans la prédiction augmente les complexités en temps et en espace des algorithmes. De plus il a été prouvé que chercher la structure

d'énergie minimale incluant les pseudonœuds en utilisant le modèle des plus proches voisins est un problème NP-difficile [135], de même que pour un modèle d'énergie plus simple incluant seulement les hélices [200]. La complexité des algorithmes prédisant les pseudonœuds a diminué au fil des années.

La majorité des algorithmes de l'état de l'art prédisent non pas l'ensemble des pseudonœuds possibles, mais un sous-ensemble. Ces sous-ensembles sont appelés *classes* et leurs relations hiérarchiques sont représentées dans la figure 2.1 A. La plus petite classe est "PKF" pour *pseudoknot free* qui regroupe toutes les structures sans pseudonœuds.

Ces ensembles de pseudonœuds sont dépendant des algorithmes et peuvent s'intersecter. Ils ne correspondent pas à des types de pseudonœuds comme ceux décrits en section 1.1.4, figure 1.9.

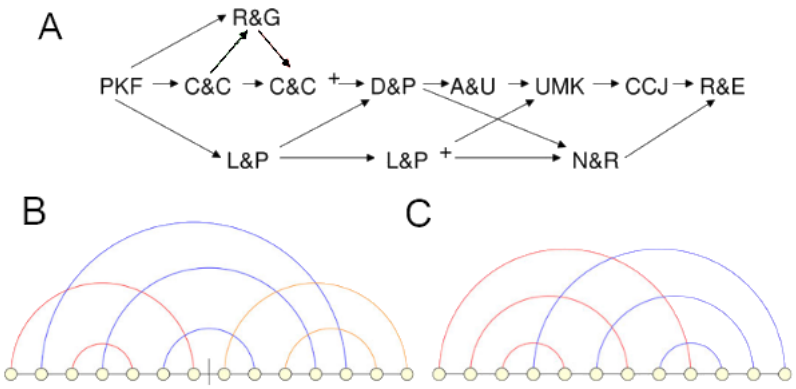

Figure 2.1 : Classes de pseudonœuds prédites par les algorithmes de prédiction de structures secondaires. A) Relations d'inclusion entre les différentes classes [153]. B) Pseudonœud représentatif de la classe A&U [5]. C) Pseudonœud représentatif de la classe UMK [220].

Nous présentons dans un premier temps les outils ne prédisant pas les pseudonœuds, puis ceux les prédisant.

2.1.1 Méthodes et outils ne prédisant pas les pseudonœuds

Les méthodes de prédiction des structures secondaires d'ARN à partir de leur séquences ont évolué avec les modèles et notamment l'amélioration des paramètres énergétiques. Grâce aux travaux réalisés sur la structure des ARN dans les années 50/60 [66], les premières méthodes basées sur la maximisation du nombre d'appariements [73] et la thermodynamique [213, 55, 214] ont vu le jour. Les méthodes de thermodynamique s'appuient sur les contributions énergétiques de certains motifs et déterminent à partir de ces contributions l'énergie libre d'une structure d'un ARN donné.

Nous présentons tout d'abord les méthodes et outils basés sur le modèle MFE, car c'est le premier modèle utilisé historiquement. Puis nous présentons les méthodes et outils basés sur le modèle MEA et plus largement sur des méthodes probabilistes.

Les méthodes et outils disponibles ne prédisant pas les pseudonœuds sont résumés dans le tableau 2.1.

Référence	Modèle	Outil	Structures sous-optimales
Nussinov et al., 1978 [157]	Maximisation du nombre d'appariements		
Waterman et Smith, 1978 [231]	MFE		
Zuker, 1989 [255]	MFE	Mfold	✓
Morgan et Higgs, 1998 [152]	Échantillonnage		✓
Wuchty et al., 1999 [242]	MFE	AllSub	✓
Mathews et al., 2004 [140]	MFE	Fold	
Lu et al., 2009 [134]	MEA	MaxExpect	✓
Ding et Lawrence, 2003 [58]	Échantillonnage	stochastic	✓
Ding et Lawrence, 2003 [58]	Échantillonnage	Sfold	✓
Do et al., 2006 [65]	MEA	CONTRAfold	✓
Dawson et al., 2006 [50]	MFE	VSfold	✓
Markham et Zuker, 2008 [136]	MFE	UNAFold	✓
Reeder et Giegerich, 2009 [177]	MFE	RNAshapes	✓
Lorenz et al., 2011 [130]	MFE, MEA	RNAfold	
Lorenz et al., 2011 [130]	MFE, MEA	RNAsubopt	✓
Jiang et al., 2019 [103]	MFE	DPARSS	✓

Tableau 2.1 : Résumé de l'état de l'art des outils de prédiction de structures secondaires d'ARN sans pseudonœuds. Les outils en gris ne sont plus ou pas encore disponible, ou ils sont obsolètes.

Méthodes et outils utilisant le modèle MFE

En 1978, Nussinov et al. présentent le premier algorithme récursif, basé sur la programmation dynamique, permettant de prédire une structure secondaire maximisant le nombre maximum d'appariements entre ses nucléotides [157]. Sa complexité en temps est $\mathcal{O}(n^3)$, où n est la longueur de la séquence nucléotidique. Une seconde version de l'algorithme est développée en 1980 et utilise les valeurs d'énergie libre des appariements [156]. En 1978 également, Waterman et Smith développent un algorithme permettant de prédire la structure d'énergie minimale d'un ARN [231]. Ces deux algorithmes permettent de trouver une structure secondaire pour un ARN donné, mais pas de structures sous-optimales.

Dans les années 80, un outil appelé Mfold, a été développé [255, 259] : il permet de prédire la structure d'énergie minimale d'un ARN avec un algorithme récursif et utilise de nombreux paramètres énergétiques déterminés à l'origine par Turner et al. [219]. Les paramètres énergétiques ont par la suite été affinés et l'algorithme a été amélioré pour prédire des structures sous-optimales [256]. Mfold est disponible en tant que webserver [257]. En 1989, Jaeger et al. améliorent les prédictions de Mfold en modifiant l'algorithme [100]. L'outil Mfold n'est maintenant plus supporté et est remplacé par l'outil UNAFold [136].

En 1994, l'équipe de l'Université de Vienne présente un ensemble logiciel, appelé ViennaRNA, qui inclut des outils de prédiction de structures secondaires [93]. Parmi ces outils, un outil appelé RNAfold est basé sur l'algorithme de Mfold. Une seconde version de RNAfold, appelée RNAsubopt, est développée pour permettre de générer des structures sous-optimales selon un intervalle d'énergies donné.

En 1994 est également développé un algorithme [225] sur lequel se base un outil de repliement d'ARN, appelé Fold [140]. Cet outil est intégré à l'ensemble logiciel RNAstructure [139] qui est développé par l'équipe de l'Université de Rochester. Une version de l'outil Fold, appelée Allsub, est développée pour permettre de générer des structures sous-optimales. Il s'agit d'une implémentation de l'algorithme de Wuchty [242].

En 1998, un nouvel algorithme est développé afin d'échantillonner des structures secondaires. Morgan et Higgs développent cet algorithme d'échantillonnage [152] à partir du modèle d'énergie de Nussinov (maximisation du nombre d'appariements).

En 2006, un outil appelé VSfold [50], utilise un nouveau modèle de calcul de l'énergie libre pour les interactions longues distances. Le modèle se base sur la somme cumulée de l'énergie des appariements. L'énergie est pondérée de manière inversement proportionnelle à la longueur de l'appariement dans la séquence de l'ARN. Ce modèle a ensuite été amélioré à plusieurs reprises [51, 52, 49] et permet de trouver des structures sous-optimales.

En 2009, Reeder et Giegerich développent un ensemble logiciel nommé RNAshapes [177]. Il rassemble des outils de prédiction de structures secondaires basés sur le modèle thermodynamique. Contrairement aux autres outils, les structures prédites qui sont similaires sont rassemblées pour éviter la redondance dans les résultats.

En 2019, un nouvel algorithme [103], appelé DPARSS, est proposé. Il permet d'explorer plus largement l'espace de recherche de structures secondaires et est performant pour les ARNt.

Méthodes et outils probabilistes : MEA et autres

En 1990, McCaskill développe un algorithme permettant de calculer la fonction de partition d'une séquence, ce qui permet de calculer les probabilités d'appariement de la séquence (voir les définitions en section 1.3.4). Grâce au calcul des probabilités d'appariements, la structure maximisant la précision attendue peut être calculée, c'est le modèle MEA [65].

CONTRAfold, développé en 2006, est un outil basé sur un modèle probabiliste appelé modèle log-linéaire conditionnel [65]. Ce modèle apprend à partir d'un jeu de données d'entraînement des probabilités conditionnelles d'appariements. Pour la prédiction d'une structure secondaire, ce modèle vise à maximiser la précision attendue.

Un outil de l'ensemble logiciel RNAstructure, appelé MaxExpect [134], permet d'utiliser le modèle MEA pour prédire la structure secondaire d'un ARN. MaxExpect permet de générer des structures sous-optimales. Cet outil permet de montrer que le modèle MEA est très performant, et qu'il permet d'obtenir de meilleures prédictions que le modèle historique MFE.

Un outil de ViennaRNA [130], appelé RNAfold, permet de calculer la fonction de partition, grâce à l'algorithme de McCaskill [144], pour calculer la structure secondaire d'un ARN selon le modèle MEA. Cependant l'outil ne permet pas de générer de structures sous-optimales.

En 2003, Ding et Lawrence mettent en avant le fait qu'un ARN peut se replier en différentes conformations. Ils développent un algorithme [58] qui calcule un ensemble de structures appelé ensemble de Boltzmann. L'algorithme échantillonne les structures à partir des probabilités d'appariement calculées grâce à la fonction de partition. L'outil résultant de cet algorithme est appelé Sfold et est disponible en tant que webserver [59]. L'algorithme est également implémenté dans un outil, appelé stochastic, de RNAstructure [139], ainsi que dans l'outil RNAsubopt de ViennaRNA en utilisant l'option -p.

2.1.2 Méthodes et outils prédisant les pseudonœuds

L'état de l'art de la prédiction de structures secondaires incluant des pseudonœuds est très diversifié.

Comme précédemment, nous présentons tout d'abord les méthodes et outils basés sur le modèle MFE, puis ceux basés sur les méthodes probabilistes et enfin ceux basés sur d'autres méthodes. Les méthodes et outils prenant en compte les pseudonœuds sont résumés dans le tableau 2.2.

Méthodes et outils utilisant le modèle MFE

Les méthodes suivantes sont toutes basées sur des algorithmes de programmation dynamique et pour la plupart minimisent l'énergie libre. Rivas et Eddy [184] ont développé à la fin des années 90 un algorithme de complexité en temps $\mathcal{O}(n^6)$ permettant de prédire des structures avec pseudonœuds. Ils ont utilisé les paramètres énergétiques standards du modèle thermodynamique ainsi que des paramètres spécifiques aux pseudonœuds. Leur méthode a donné lieu à un outil appelé PKNOTS. La classe de pseudonœuds prédite par l'outil PKNOTS est notée "R&E", c'est la plus générale, elle regroupe tous les pseudonœuds car l'algorithme permet de déterminer tous les repliements possibles d'un ARN. Cependant, cet outil ne permet pas de générer de structures sous-optimales et n'est plus disponible actuellement.

La même année, Uemura et al. [220] développent un algorithme basé sur une grammaire fonctionnant avec des ensembles d'arbres, appelée grammaire d'arbres adjoints. Les arbres sont ensuite sélectionnés selon l'énergie de leur structure secondaire associée. L'algorithme a une complexité en temps de $\mathcal{O}(n^5)$ et permet de déterminer des structures de la classe UMK (voir figure 2.1 B).

Akutsu et Uemura [5] ont présenté un algorithme de complexité $\mathcal{O}(n^4)$ en temps, basé sur la maximisation du nombre d'appariements. La classe de pseudonœuds prédite par cet algorithme est notée "A&U", elle rassemble des pseudonœuds où les bases de deux hélices sont intercalées arbitrairement (voir figure 2.1 C).

La même année, Lyngsø et Pedersen [135] développent un algorithme qui utilise le même modèle thermodynamique que Rivas et Eddy. La classe de pseudonœuds prédite est notée "L&P$^+$" et rassemble des structures de la forme

Référence	Modèle	Outil	Structures sous-optimales
Rivas et Eddy, 1999 [184]	MFE	PKNOTS	
Uemura et al., 1999 [220]	MFE		
Akutsu, 2000 [5]	Maximisation du nombre d'appariements		
Lyngsøet Pedersen, 2000 [135]	Maximisation du nombre d'appariements		
Dirks et Pierce, 2003 [62]	MFE		
Ruan et al., 2004 [186]	MFE	ILM	
Reeder et Giegerich, 2004 [176]	MFE	pknotsRG	✓
Ren et al., 2005 [182]	MFE	Hotknots	
Chen et al., 2008 [39]	MFE	Flexstem	
Parisien et Major, 2008 [164]	MFE	MC-Fold	✓
Metzler et Nebel, 2008 [146]	Échantillonnage		✓
Cao et Chen, 2009 [34]	MFE		
Chen et al., 2009 [38]	MFE		
Poolsap et al., 2009 [167]	MFE		
Bindewald et al., 2010 [22]	MFE	CyloFold	
Bellaousov et Mathews, 2010 [16]	MEA	ProbKnot	
Reidys et al., 2011 [179]	MFE, Échantillonage	gfold	
Lorenz et al., 2011 [130]	MFE	RNAPKplex	
Sato et al., 2011 [194]	MEA	IPknot	
Bon et Orland, 2011, [29]	MFE	TT2NE	
Bon et al., 2012 [30]	Échantillonnage	McGenus	✓
Janssen et Giegerich, 2014 [101]	MFE	pKiss	✓
Saule et Giegerich, 2015 [195]	MFE et MEA		✓
Dallaire et Major, 2016 [46]	MFE	MC-FlashFold	✓
Jabbari et al., 2018 [98]	MFE	Knotty	
Wang et al., 2019 [228]	Apprentissage profond	DMfold	

Tableau 2.2 : Résumé de l'état de l'art des outils de prédiction de structures secondaires d'ARN avec pseudonœuds. Les outils en gris ne sont plus ou pas encore disponible, ou ils sont obsolètes.

$w_1 u_1 w_2 u_2 w_3$, où w_i et u_j sont des hélices et où $w_1 w_2 w_3$ et $u_1 u_2$ sont des structures sans pseudonœuds. La complexité en temps de l'algorithme est de $\mathcal{O}(n^5)$. La sous-classe notée "L&P" rassemble des structures de la forme $w_1 u_1 w_2 u_2$, où $w_1 w_2$ et $u_1 u_2$ sont des structures sans pseudonœuds, qui peuvent être trouvées en $\mathcal{O}(n^3)$.

Lyngsø et Pedersen [135] montrent également que prédire la structure secondaire ayant l'énergie libre minimum, avec des pseudonœuds, est un problème NP-complet.

Dirks et Pierce [62] ont développé en 2003 un algorithme permettant de calculer non seulement la structure d'énergie libre minimale avec pseudonœuds, mais aussi la fonction de partition. Leur algorithme (de complexité en temps

$\mathcal{O}(n^5)$) permet de trouver des pseudonœuds de type H (voir figure 1.9). Cette classe de pseudonœuds est notée D&P. Cet algorithme est implémenté dans un ensemble logiciel, appelé NUPACK [248]. Leur algorithme permet également de générer des structures sous-optimales.

En 2004, un outil appelé ILM, est développé par Ruan et al. [186], il utilise l'énergie libre et/ou la covariance pour déterminer des structures secondaires avec tout type de pseudonœuds. L'algorithme est une extension de l'algorithme de *loop matching* [157] pour inclure la prédiction des pseudonœuds. ILM est capable d'utiliser soit la minimisation de l'énergie, soit l'approche comparative ou les deux.

En 2009, Cao et Chen [34] ont développé un algorithme de complexité $\mathcal{O}(n^6)$ prédisant des pseudonœuds de type H ayant des hélices dans leurs boucles qui correspondent à deux classes : C&C et C&C$^+$. La même année, Chen et al. [38] ont développé un algorithme de complexité en temps $\mathcal{O}(n^5)$ prédisant les pseudonœuds de type H et HHH. La classe de prédiction de pseudonœuds de cet algorithme est notée CCJ, les pseudonœuds sont formés par deux des structures de la classe UMK, appelées *TGB gapped structures* (voir figure 2.1 C), avec un *gap* à la position indiquée.

Plus récemment en 2018 a été proposé une amélioration de l'algorithme proposé par Chen et al. Le nouvel algorithme appelé Knotty [98] permet de trouver plus de types pseudonœuds et d'évaluer l'énergie des structures de manière réaliste.

En 2011, Reidys et al. [179] ont développé un algorithme basé sur une grammaire et sur la classification topologique des pseudonœuds pour déterminer la structure d'énergie libre minimale d'un ARN. L'outil résultant de cette méthode est appelé gfold.

HotKnots [182] est un outil développé en 2005 utilisant une heuristique. Cette heuristique est basée sur une recherche itérative des hélices stables en utilisant un algorithme de minimisation d'énergie trouvant des structures sans pseudonœuds. Un outil de ViennaRNA [130], appelé RNAPKplex, est également capable de prédire des structures secondaires avec pseudonœuds grâce à une heuristique. L'algorithme de complexité en temps $\mathcal{O}(n^4)$ calcule des intervalles de structures non-appariées et accessibles, puis calcule l'énergie si les deux intervalles forment une hélice.

Un outil appelé FlexStem [39], utilise une heuristique basée sur la minimisation d'énergie et le mécanisme de repliement des ARN. La méthode simule le processus de repliement en ajoutant successivement des hélices de longueur maximale. Un autre outil, appelé CyloFold [22], cherche lui aussi à simuler le processus de repliement en choisissant des hélices stables pour prédire des structures secondaires avec pseudonœuds.

En 2004, Reeder et Giegerich [176] ont présenté un algorithme de complexité $\mathcal{O}(n^4)$ permettant de prédire des pseudonœuds de la classe notée R&G. C'est une classe particulière de pseudonœuds de type H, appelés canoniques récursifs simples, où les extrémités 5' et 3' des hélices doivent avoir la même longueur et ne doivent pas être interrompues par des boucles internes. Les hélices doivent également être saturées : si des appariements sont possibles, ils doivent exister. L'outil implémentant l'algorithme permettant de trouver

ces structures est appelé pknotsRG. Cet algorithme a été amélioré en 2015 par Janssen et al. [101], avec un outil appelé pKiss, qui prédit des pseudonœuds de type HHH (voir figure 1.9). L'outil pKiss prédit donc plus de pseudonœuds que pknotsRG qui peut seulement prédire certains types H. pknotsRG et pKiss peuvent générer plusieurs structures sous-optimales.

Un outil, appelé MC-Fold [164] et développé en 2008, introduit un nouveau modèle de prédiction dans lequel les relations entre les motifs sont représentées par des motifs cycliques nucléotidiques. Ces motifs permettent d'avoir plus d'information autour d'un nucléotide et d'établir une nouvelle fonction d'évaluation des structures. MC-Fold permet de trouver des pseudonœuds de type H, ainsi que des appariements non-canoniques, et de retourner plusieurs structures secondaires. Cet outil a été amélioré en 2016, donnant lieu à un outil plus rapide : MC-Flashfold [46].

En 2011, Bon et Orland développent un outil appelé TT2NE [29] permettant de trouver des structures avec pseudonœuds selon l'énergie et le genre des structures secondaires. Le genre traduit la complexité d'un pseudonœud mais est différent de la profondeur (voir définition 1.1.1) ou des différents types de pseudonœuds. Le genre (noté g) se calcule suivant le nombre d'appariements (P) et de boucles (L) d'une structure, de la manière suivante : $g = \frac{P-L}{2}$. Leur outil implémente un algorithme exacte permettant de calculer le genre. En 2012, Bon et Orland présentent McGenus [30] qui est un outil basé sur un algorithme de Monte Carlo et qui cherche les structures ayant un score minimum comprenant l'énergie libre et le genre. L'outil McGenus effectue une recherche stochastique qui permet de trouver différents types de pseudonœuds. Il permet de générer des structures sous-optimales.

La plupart des outils cités précedemment utilisent des algorithmes basés sur la programmation dynamique. Cependant, la programmation linéaire a également été utilisée pour modéliser le problème de repliement des ARN avec pseudonœuds. En 2009, Poolsap [167] a proposé un programme linéaire minimisant l'énergie libre qui permet de prédire les pseudonœuds de type H avec des hélices à l'intérieur de leurs boucles. Le programme linéaire proposé ne permet pas d'obtenir de structures sous-optimales.

Méthodes et outils probabilistes : MEA et autres

En 2008, Metzler et Nebel [146] développent un algorithme d'échantillonnage permettant de générer des structures avec tous types de pseudonœuds. Leur algorithme est basé sur la méthode de Monte-Carlo des chaines de Markov leur permettant d'échantillonner les structures selon une distribution calculée grâce à une grammaire hors-contexte probabiliste (leur permettant de générer des structures sans pseudonœud) combinée avec la possibilité d'ajouter des hélices arbitrairement (ajout de pseudonœuds, basée sur la combinaison de Cai et al. [32]).

En 2010, Bellaousov et al. [16] développent un outil, appelé ProbKnot, inclus dans RNAstructure. Cet outil utilise un algorithme qui maximise la précision attendue et ayant une complexité en temps égale à $\mathcal{O}(n^3)$. Le principe de cet algorithme est simple : il calcule les probabilités d'appariements grâce à l'algorithme de McCaskill, puis assemble des structures sans pseudonœuds

prédites à partir des probabilités. Cela permet de prédire tous les types de pseudonœuds. Cependant, cet outil ne prédit pas de structures sous-optimales.

En 2011, Sato et al. [194] ont développé un outil, appelé IPknot, qui utilise un programme linéaire maximisant la prédiction attendue pour prédire les pseudonœuds. IPknot est aussi capable de prédire des structures consensus avec pseudonœuds en utilisant les informations d'évolution et des probabilités d'appariements. IPknot ne prédit pas de structures sous-optimales.

L'outil gfold [179], présenté précédemment, peut également faire de l'échantillonnage en calculant la fonction de partition.

Autres méthodes

Une méthode bi-objectif en programmation dynamique [195] a été développée en 2015 pour combiner les deux modèles MFE et MEA. Cependant, aucun outil n'est disponible.

Très récemment en 2019, Wang et al. ont développé une méthode appelée DMfold [228], basée sur l'apprentissage profond et la maximisation du nombre d'appariements. Cette méthode prédit dans un premier temps des structures sans pseudonœuds, qui sont assemblées, pour prédire dans un second temps les pseudonœuds. L'apprentissage est réalisé sur les séquences et les structures en format parenthésé de diverses familles comprenant presque 4000 ARN, dont certains avec pseudonœuds.

Outils utilisés dans nos expérimentations

Les outils les plus intéressants pour nous sont ceux pouvant prédire les pseudonœuds et pouvant retourner des solutions sous-optimales. Parmi l'état de l'art que nous venons de présenter, seuls pKiss [101], McGenus [30], MC-Fold [164] et MC-Flashfold [46] répondent à ces critères.

— Le principe de pKiss est de décomposer les séquences d'ARN en toutes les sous-séquences possibles avec une grammaire. Ensuite, la structure secondaire d'énergie libre minimale est calculée à partir des décompositions. Pour réduire l'espace de recherche, pKiss est basé sur des règles canoniques qui réduisent le nombre de pseudonœuds (seulement certains pseudonœuds de type H et les pseudonœuds de type HHH). Grâce à un algorithme non-ambigu de programmation dynamique, la redondance est également réduite.
— McGenus est basé sur un algorithme de Monte Carlo qui permet de trouver la structure ayant un score minimum. Ce score inclut l'énergie libre et le genre de la structure qui exprime la complexité d'un pseudonœud. L'outil McGenus effectue une recherche stochastique qui permet de trouver divers types de pseudonœuds.
— MC-Fold prédit des structures secondaires avec des pseudonœuds de type H et des appariements non-canoniques. Il utilise un modèle dans lequel les ARN sont modélisés sous forme de graphes qui sont décomposés en de nombreux cycles.
— MC-Flashfold [46] est une amélioration de l'outil MC-Fold, son temps d'exécution est nettement plus rapide.

Dans nos expérimentations, nous considérons aussi l'outil RNAsubopt [130], un outil performant pour la prédiction de structures sans pseudonœuds, permettant de prédire des structures sous-optimales. Cela nous permet d'évaluer notre outil BiokoP, sur la prédiction de structures sans pseudonœuds.

2.2 Notre méthode : BiokoP

Nous présentons ici une nouvelle méthode de prédiction de structures secondaires d'ARN, appelée Biokop, permettant de prédire les pseudonœuds et de retourner plusieurs solutions optimales et sous-optimales.

Parmi les outils disponibles, peu sont capables de prédire tous les types de pseudonœuds. En effet, la plupart des outils sont basés sur un algorithme récursif de programmation dynamique pour trouver la structure d'énergie minimum. Pour trouver tous les types de pseudonœuds possibles, les récursions de cet algorithmes doivent être modifiées, rendant l'algorithme plus complexe. Dans Biokop, nous utilisons la programmation linéaire qui, de par sa flexibilité, nous permet de prédire tous les types de pseudonœuds, sans augmenter la complexité de conception du programme linéaire.

La programmation linéaire nous permet par ailleurs de combiner deux modèles de prédiction, les modèles MFE et MEA. La combinaison de ces deux modèles permet de mieux approcher les structures réelles. Or, comme nous l'avons vu dans la littérature, il n'existe à notre connaissance aucun outil réalisant cette combinaison (il existe une méthode bi-objectif en programmation dynamique publiée en 2015 [195] mais sans outil associé).

Nous proposons ici un outil nommé BiokoP (*Bi-objective pseudoknot prediction*) pour prédire des structures secondaires d'ARN, optimales et sous-optimales, à partir d'une séquence d'ARN.

Dans cette section, nous présentons tout d'abord, les programmes linéaires en nombres entiers utilisés dans la littérature pour prédire des structures secondaires d'ARN pouvant inclure des pseudonœuds. Puis décrivons notre programme linéaire en nombres entiers bi-objectif combinant les modèles MFE et MEA. Enfin, nous présentons un nouvel algorithme générique permettant de résoudre des programmes linéaires en nombres entiers bi-objectifs.

2.2.1 Un programme linéaire bi-objectif combinant les modèles MEA et MFE

Dans cette section nous décrivons comment nous modélisons le problème de repliement de l'ARN avec la programmation linéaire bi-objectif (voir rappels sur la programmation linéaire en section 1.2.3, page 26). Nous décrivons les deux objectifs auxquels nous nous référons par Mod1 et Mod2. Le premier objectif, Mod1, correspond à la maximisation de la précision attendue (MEA) et est basé sur le programme linéaire développé par Sato et al. [194]. Le deuxième objectif, Mod2, correspond à la minimisation de l'énergie libre (MFE) et est basé sur le programme linéaire développé par Poolsap [167]. Ces deux modèles peuvent générer tout type de pseudonœuds. Nous notons Mod1^{so} et Mod2^{so}

(*so* pour sous-optimal) les extensions de Mod1 et Mod2, respectivement, retournant plusieurs solutions sous-optimales.

Les variables

Dans Mod1 et Mod2, les structures secondaires sont modélisées de la manière suivante. Une séquence d'ARN s est composée de n nucléotides (A, U, G ou C). Chaque base peut être appariée suivant les appariements Watson-Crick (A-U et G-C) ou Wobble (G-U). Pour prendre en compte les pseudonœuds, on suppose qu'une structure secondaire peut être décomposée en m sous-structures sans pseudonœuds y^1, y^2,..., y^m, appelées niveaux. La profondeur d'un pseudonœud correspond au nombre minimum de niveaux nécessaire pour le décomposer. Les niveaux sont des ensembles disjoints, c'est-à-dire qu'un appariement appartient exactement à un seul niveau. À partir des données expérimentales, il est généralement supposé que deux niveaux suffisent à décrire la plupart des structures d'ARN. Dans la suite, nous considérons donc que $m = 2$. Une structure sans pseudonœuds a un seul niveau, $m = 1$.

Un appariement entre deux bases i et j existe au niveau y^p si la variable binaire y^p_{ij} est égale à 1, avec $i = 1,\ldots,n$ et $j = i+1,\ldots,n$. Sinon la variable est égale à 0. Les appariements possibles sont répartis en 6 types numérotés de 1 à 6, correspondant respectivement aux appariements suivant : A-U, C-G, G-C, G-U, U-G et U-A. Pour calculer l'énergie selon les paramètres de paires d'appariements consécutifs, des variables binaires supplémentaires sont définies. Soient (i, j) et $(i-1, j+1)$ deux appariements possibles consécutifs d'un niveau y^p. La variable binaire correspondante est notée x^{klp}_{ij}, avec k (respectivement l) le type de l'appariement (i, j) (respectivement $(i-1, j+1)$). Si x^{klp}_{ij} est égale à 1, cela signifie que les bases i et j sont appariées ensemble ainsi que les bases $i-1$ et $j+1$. En revanche si x^{klp}_{ij} est égale à 0, cela signifie soit qu'un seul appariement existe, ou bien aucun des deux.

Les fonctions objectifs et les contraintes

Dans notre problème linéaire, nous combinons les modèles Mod1 et Mod2. L'objectif de Mod1 est de maximiser la précision attendue. Afin d'exprimer ce premier objectif, il est nécessaire tout d'abord de calculer les probabilités d'appariement avec le modèle de Dirks et Pierce [62]. Soit p_{ij} la probabilité d'appariement entre les nucléotides i et j. Nous notons f_1 la fonction approximant la précision attendue :

$$f_1(y) = \sum_{1\leq p\leq m} \beta^p \sum_{i<j \text{ t.q. } p_{ij}>\theta^p} p_{ij} y^p_{ij}, \tag{2.1}$$

où β^p sont des constantes définies pour tous les niveaux p (fixées à $\beta^p = 1/m$). θ^p est un seuil visant à ignorer les probabilités les plus basses. Suivant ce seuil, un grand nombre de probabilités peuvent être éliminées, rendant caduques les variables de décision associées. Ces dernières ne sont alors pas créées, ce qui réduit la taille du programme linéaire. Ce seuil est fixé à $1/(\gamma^p+1)$ pour Mod1,

avec γ^p un poids dépendant du niveau, fixé à 2 pour le premier niveau et à 4 pour le second niveau.

L'objectif de Mod2 est de minimiser l'énergie libre. Pour cela, la fonction d'énergie consiste à sommer les énergies de chaque paire d'appariements consécutifs :

$$f_2(x) = \sum_{p=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{6} \sum_{l=1}^{6} e_{kl} x_{ij}^{klp}, \tag{2.2}$$

où e_{kl} est l'énergie utilisée par le programme linéaire développé par Poolsap [167] (voir tableau 2.3).

	A-U	C-G	G-C	G-U	U-G	U-A
A-U	-1.1	-2.1	-2.2	-1.4	-0.9	-0.6
C-G	-2.1	-2.4	-3.3	-2.1	-2.1	-1.4
G-C	-2.2	-3.3	-3.4	-2.5	-2.4	-1.5
U-G	-0.9	-2.1	-2.4	-1.3	-1.3	-1.0
U-A	-0.6	-1.4	-1.5	-0.5	-1.0	-0.3

Tableau 2.3 : Paramètres énergétiques des paires d'appariements utilisés dans le programme linéaire de Poolsap [167].

Le programme linéaire bi-objectif est défini de la manière suivante :

$$\max f_1(x)$$
$$\min f_2(x)$$

tel que :

$$x_{ij}^{klp} \leq \frac{y_{ij}^p + y_{i+1,j-1}^p}{2} \qquad 1 \leq i \leq n-1, 2 \leq j \leq n \tag{2.3}$$

Ces contraintes permettent d'assurer la cohérence entre les variables de paires d'appariements consécutifs x_{ij}^{klp} et les variables d'appariements y_{ij}^p : si deux appariements se suivent alors la variable associée à ces deux appariements peut être égale à 1 et doit être égale à 0 dans le cas contraire.

$$\sum_{1 \leq p \leq m} \left\{ \sum_{h=1}^{i-1} y_{hi}^p + \sum_{h=i+1}^{n} y_{ih}^p \right\} \leq 1 \qquad 1 \leq i \leq n \tag{2.4}$$

Ces contraintes empêchent chaque base i d'être appariée avec plusieurs autres bases (voir figure 2.2 A) : la somme de toutes les variables d'appariements y associées à un nucléotide i doit être égale à 1 au maximum.

$$y_{ij}^p + y_{kl}^p \leq 1 \qquad 1 \leq p \leq m, 1 \leq i < k < j < l \leq n \tag{2.5}$$

Ces contraintes forcent chaque sous-structure à être sans pseudonœuds (voir figure 2.2 B) : deux variables d'appariements ne peuvent être égales à 1 simultanément si elles sont sur le même niveau y^p et si les appariements se croisent.

$$\sum_{i<k<j<l} y_{ij}^q + \sum_{k<i'<l<j'} y_{i'j'}^q \geq y_{kl}^p \qquad 1 \leq q < p \leq m, 1 \leq k < l \leq n \tag{2.6}$$

Ces contraintes forcent l'utilisation des niveaux inférieurs pour établir des appariements : si des appariements sont disposés comme en figure 2.2 C, la somme des variables d'appariement y ayant le niveau le plus petit doit être supérieure à la variable d'appariement ayant le plus haut niveau. Ces contraintes permettent également d'utiliser un nombre minimum de niveaux.

$$l^p_{i-1} + (1 - l^p_i) + l^p_{i+1} \geq 1 \qquad \text{où } l^p_i = \sum_{j=i+1}^{n} y^p_{ij}$$

$$1 \leq p \leq m, 1 < i \leq n \tag{2.7}$$

$$r^p_{i-1} + (1 - r^p_i) + r^p_{i+1} \geq 1 \qquad \text{où } r^p_i = \sum_{j=1}^{i-1} y^p_{ji}$$

$$1 \leq p \leq m, 1 < i \leq n \tag{2.8}$$

Ces contraintes empêchent les appariements isolés : les appariements consécutifs sont favorisés.

$$y^p_{ij} = 0 \qquad 1 \leq p \leq m, 1 \leq i, j \leq n \quad | \quad |j - i| < 4 \tag{2.9}$$

Ces contraintes empêchent les boucles d'une longueur inférieure à 3 nucléotides.

$$x^{klp}_{ij} \in \{0,1\} \qquad \text{avec} \qquad 1 \leq p \leq m, 1 \leq i \leq n-1, 2 \leq j \leq n, 1 \leq k, l \leq 6$$

$$y^p_{ij} \in \{0,1\} \qquad \text{avec} \qquad 1 \leq p \leq m, 1 \leq i, j \leq n, 1 \leq k, l \leq 6$$

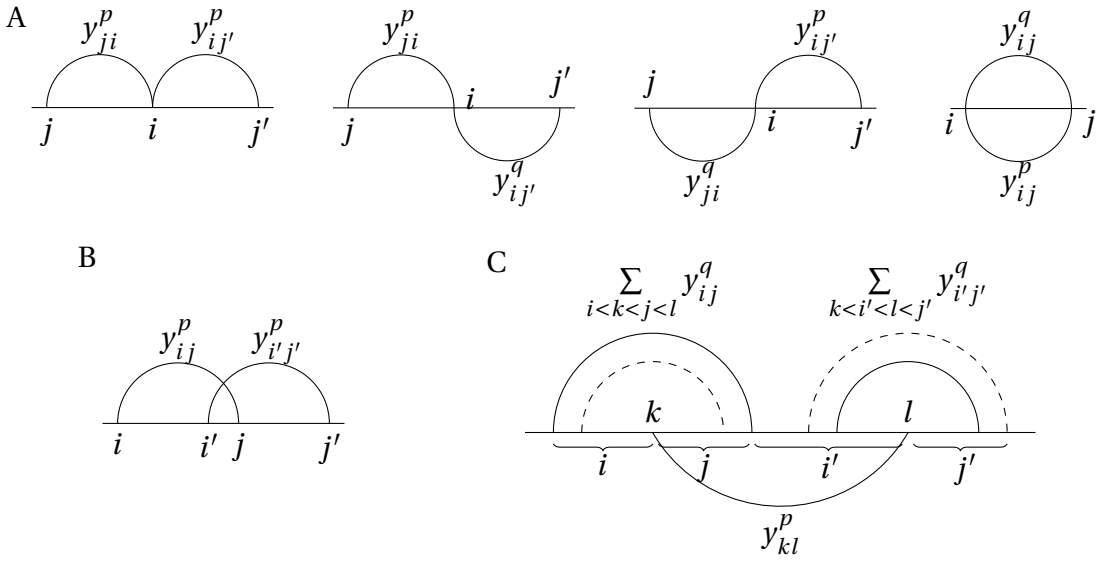

Figure 2.2 : Illustration des contraintes 2.4 (A), 2.5 (B) et 2.6 (C).

2.2.2 Nouvelle méthode de résolution de programmes bi-objectifs

Dans cette section, nous présentons un nouvel algorithme générique déterminant les k meilleurs ensembles de Pareto d'un programme linéaire en nombres entiers bi-objectif (voir rappels sur l'optimisation multi-objectif en section 1.2.3, page 27). À notre connaissance, aucune méthode de ce type n'existait jusqu'à présent dans la littérature. Le principe de notre algorithme est de résoudre un programme linéaire mono-objectif plusieurs fois en effectuant une recherche dichotomique sur l'espace des solutions.

Le type de programme linéaire bi-objectif à résoudre

Les programmes linéaires bi-objectifs que notre algorithme peut résoudre sont les suivants :

$$\min f_1(x)$$
$$\min f_2(x)$$

tel que :

$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$
$$x = (x_1, x_2, \ldots, x_n)$$
$$x_i \in \mathbb{Z} \qquad 1 \leq i \leq n$$

Les contraintes sont définies ici comme des fonctions linéaires g_j en fonction des variables de décisions $x_1, \ldots, x_n$.

Par simplicité, nous supposons ici que les variables de décisions sont binaires. Nous avons alors, étant donné un ensemble de solutions interdites F, le programme générique suivant, noté $P(\lambda^-, \lambda^+, F)$:

$$\min f_1(x)$$

tel que :

$$f_2(x) \geq \lambda^-$$
$$f_2(x) \leq \lambda^+$$
$$\text{DIFF}(s) \forall s \in F$$
$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$
$$x = (x_1, x_2, \ldots, x_n)$$
$$x_i \in \mathbb{Z} \qquad 1 \leq i \leq n$$

Dans ce programme P, la fonction objectif correspond à la première fonction objectif du programme bi-objectif original, tandis que la deuxième fonction objectif est exprimée à travers deux contraintes maintenant sa valeur entre deux seuils : λ^- et λ^+. Le signe de la fonction f_2 du programme est inversé pour avoir deux objectifs à minimiser.

Dans le programme générique P, pour chaque solution s de F, une contrainte notée DIFF(s) [14] est ajoutée. Cette contrainte interdit de retrouver la solution s. La contrainte est définie de la manière suivante. Supposons

que nous avons précédemment trouvé une solution au programme P, notée $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$. Cette solution se trouve dans l'ensemble F. La contrainte $\text{DIFF}(x^*)$ est telle que : $\sum_{i \text{ t.q } x_i^*=1}(1-x_i) + \sum_{i \text{ t.q } x_i^*=0} x_i \geq 1$. Cette contrainte assure que la distance de Hamming entre chaque solution possible s et la solution interdite x^* est d'au moins de 1. Ainsi, au moins une variable de décision x_i doit prendre une valeur différente de x_i^*.

Dans le programme linéaire bi-objectif à variables binaires de la section précédente, pour toute solution $s^* \in F$, les contraintes suivantes sont ajoutées :

$$\sum_{p=1}^{m} \sum_{ij \text{ t.q.} y_{ij}^p=1 \text{ dans } s^*} y_{ij}^p - \sum_{p=1}^{m} \sum_{ij \in \text{ t.q.} y_{ij}^p=0 \text{ dans } s^*} y_{ij}^p \leq (\sum_{p=1}^{m} |\{y_{ij}^p | y_{ij}^p = 1 \text{ dans } s^*\}|) - 1 \qquad \forall s \in F \quad (2.10)$$

Ainsi que les contraintes suivantes :

$$\sum_{ij \text{ t.q.} y_{ij}^2=1 \text{ dans } s^*} y_{ij}^1 + \sum_{ij \text{ t.q.} y_{ij}^1=1 \text{ dans } s^*} y_{ij}^2 - \sum_{ij \text{ t.q.} y_{ij}^2=0 \text{ dans } s^*} y_{ij}^1 - \sum_{ij \text{ t.q.} y_{ij}^1=0 \text{ dans } s^*} y_{ij}^2 \leq (|\{y_{ij}^1 | y_{ij}^1 = 1 \text{ dans } s^*\}| + |\{y_{ij}^2 | y_{ij}^2 = 1 \text{ dans } s^*\}|) - 1. \quad (2.11)$$

Ces dernières contraintes visent à interdire des solutions que nous appelons "miroirs". En effet, dans ce problème linéaire, les appariements peuvent être répartis sur différents niveaux et en intervertissant les niveaux, nous pouvons obtenir plusieurs solutions différentes correspondant à une même structure (elles peuvent avoir des valeurs de fonctions objectifs différentes). Ces solutions miroirs apportent de la redondance, et ces contraintes permettent de les éviter.

Dans le cas d'un programme générique P en nombres entiers, plusieurs variables binaires et continues doivent être ajoutées, ainsi que plusieurs contraintes, cela est décrit dans la section 5.2.2 du chapitre 5.

Algorithme de résolution de programmes bi-objectifs

Le principe de l'algorithme est d'effectuer une recherche dichotomique de solutions au programme P en réduisant l'espace de recherche autour de chaque nouvelle solution trouvée. Soit k le nombre d'ensembles de Pareto recherchés. À la fin de l'algorithme, l'ensemble $\mathscr{R}$ contient toutes les solutions appartenant aux k meilleurs ensembles de Pareto. Chaque solution trouvée par l'algorithme possède un label, noté $l(s)$, indiquant l'indice de l'ensemble de Pareto auquel la solution appartient, c'est-à-dire, $l(s) = k$ si et seulement si s appartient au k^{e} ensemble de Pareto.

Notre algorithme, nommé *FindKParetoSets*, fonctionne de la manière suivante. Tout d'abord nous cherchons la solution la plus à gauche, notée L, minimisant la fonction objectif f_1. Son label est fixé à 1 et la solution est ajoutée à l'ensemble $\mathscr{R}$. Il peut exister plusieurs solutions minimisant f_1 avec différentes valeurs pour f_2, cette solution L n'appartient pas forcément au premier ensemble de Pareto. Dans ce cas, son label sera modifié au cours de l'exécution de l'algorithme.

Ensuite, nous calculons la solution U maximisant la fonction objectif f_2. La valeur du premier critère f_1 d'une solution s est notée $f_1(s)$ (et respectivement $f_2(s)$ pour f_2). La valeur $f_2(U)$ va servir de limite supérieure pour la recherche récursive. La procédure LocalPareto est ensuite appelée et effectue la recherche dichotomique en commençant en dessous de la solution L, entre $-\infty$ et $f_2(L) - \epsilon$, puis au dessus de L, entre $f_2(L)$ et $f_2(U)$. Ici ϵ est une constante, la plus petite possible, c'est-à-dire telle que pour toute paire de solutions $\{s, s'\}$, on a soit $f_2(s) = f_2(s')$, soit $|f_2(s) - f_2(s')| > \epsilon$.

Données : Nombre d'ensembles recherchés k.
Résultat : Les k meilleurs ensembles de Pareto exacts.

$L := \text{solve}(P(-\infty, +\infty, \emptyset))$;
$l(L) := 1$;
$\mathscr{R} := \{L\}$;
$U := solve(P_2)$;
LocalPareto$(-\infty, f_2(L) - \epsilon, \emptyset)$; // Procédure 1
LocalPareto$(f_2(L), f_2(U), \{L\})$;
retourner $\mathscr{R}$

Algorithme 1 : FindKParetoSets(k)

Données : Seuil minimum λ^-, seuil maximum λ^+, ensemble de solutions interdites F.
Résultat : Les solutions de P sur une portion (λ^-, λ^+) de l'espace de recherche.

$F := \{x \in \mathscr{R} : \lambda^- \leq f_2(x) \leq \lambda^+\}$;
$s := solve(P(\lambda^-, \lambda^+, F))$;
si $s \neq \emptyset$ **alors**
  $l(s) := 1$;
  si $\mathscr{L} := \{x \in \mathscr{R}, x \succ s\} \neq \emptyset$ **alors**
    $l(s) := \max_{x \in \mathscr{L}} l(x) + 1$;
  fin
  si $l(s) < k + 1$ **alors**
    $\mathscr{R} := \mathscr{R} \cup \{s\}$;
    si ($\exists x \in R$ *t.q.* $f_1(x) = f_1(s)$ *ET* $x \neq s$) *ET* ($\nexists x \in R$ *t.q.* $f_1(x) = f_1(s)$ *ET* $f_2(x) = f_2(s)$ *ET* $x \neq s$) **alors**
      pour $x \in \mathscr{R}$ *t.q.* $f_1(x) = f_1(s)$ *ET* $x \prec s$ **faire**
        $l(x) := l(x) + 1$;
      fin
    fin
    LocalPareto$(\lambda^-, f_2(s) - \epsilon)$ // ;
    si $l(s) \leq k$ **alors**
      LocalPareto$(f_2(s), \lambda^+)$;
    fin
  fin
fin

Procédure 1 : LocalPareto(λ^-, λ^+, F)

La procédure *LocalPareto()* correspond au calcul d'une portion d'un en-

semble de Pareto. Elle est effectuée entre deux seuils pris en entrée, λ^- et λ^+, de la fonction objectif f_2. L'ensemble F représente l'ensemble des solutions trouvées par les appels précédents et dont la valeur de la fonction f_2 se situe entre les seuils λ^- et λ^+. L'ensemble F va permettre d'interdire ces solutions comme expliqué précédemment. Si le problème $P(\lambda^-, \lambda^+, F)$ a une solution s (lignes 2-3), par défaut le label de cette solution est 1 (ligne 6). Ensuite le label de s est calculé suivant les lignes 5-7. Si le label est inférieur ou égal à $k+1$, la solution s est ajoutée à $\mathscr{R}$. Si cela est nécessaire, les labels des solutions de $\mathscr{R}$ trouvées précédemment sont mis à jour (lignes 10 à 14). Enfin, la procédure *LocalPareto()* est appelée pour continuer la recherche en dessous de la solution s (entre λ^- et $f_2(s)-\varepsilon$) et au dessus de s (entre $f_2(s)$ et λ^+) si le label est inférieur à k.

2.2.3 Outil BiokoP

Nous avons implémenté le programme linéaire présenté précédemment en C++ en utilisant l'API du solveur CPLEX Optimizer V12.6.3 [96]. Ce programme linéaire bi-objectif est résolu grâce à l'algorithme présenté précédemment et les paramètres utilisés sont fixés à : $\varepsilon = 0.001$ et $m = 2$.

Nous obtenons ainsi un outil appelé BiokoP, qui permet de prédire plusieurs structures secondaires possibles (optimales et sous-optimales) pour un ARN d'intérêt donné. Notre outil est disponible en tant que webserver sur la plateforme EvryRNA (`https://EvryRNA.ibisc.univ-evry.fr/`), dont un aperçu est disponible en figure 2.3.

2.3 Résultats

Nous présentons dans cette section une étude comparant les deux modèles MFE et MEA, ainsi que la pertinence de combiner ces deux modèles et de générer des structures sous-optimales. Nous évaluons également notre outil BiokoP et le comparons avec des outils de la littérature. Nous présentons dans un premier temps les jeux de données que nous utilisons pour nos analyses.

2.3.1 Jeux de données

Notre outil a été testé sur un ensemble de 198 structures secondaires d'ARN avec pseudonœuds, importées depuis la base de données PseudoBase++ [212]. Dans cette base de données, les structures sont classées selon le type de pseudonœud (voir figure 1.9). Notre jeu de données, noté $B_{\text{nœud}}$, rassemble 5 types de pseudonœuds : 154 H, 3 HHH, 26 HLout, 4 HLin et 11 LL. Tous les ARN de type H sont aussi présents dans le jeu de données établi par Huang et al. [95], noté $pk168$. Il comprend 168 ARN dont les séquences sont non-redondantes et dont les longueurs sont inférieures à 140 nucléotides.

Nous avons également établi un second jeu de données, à partir de la base de données RNA STRAND [11], dans le but de tester BiokoP sur des ARN ayant des structures secondaires sans pseudonœuds. Ce jeu de données, noté

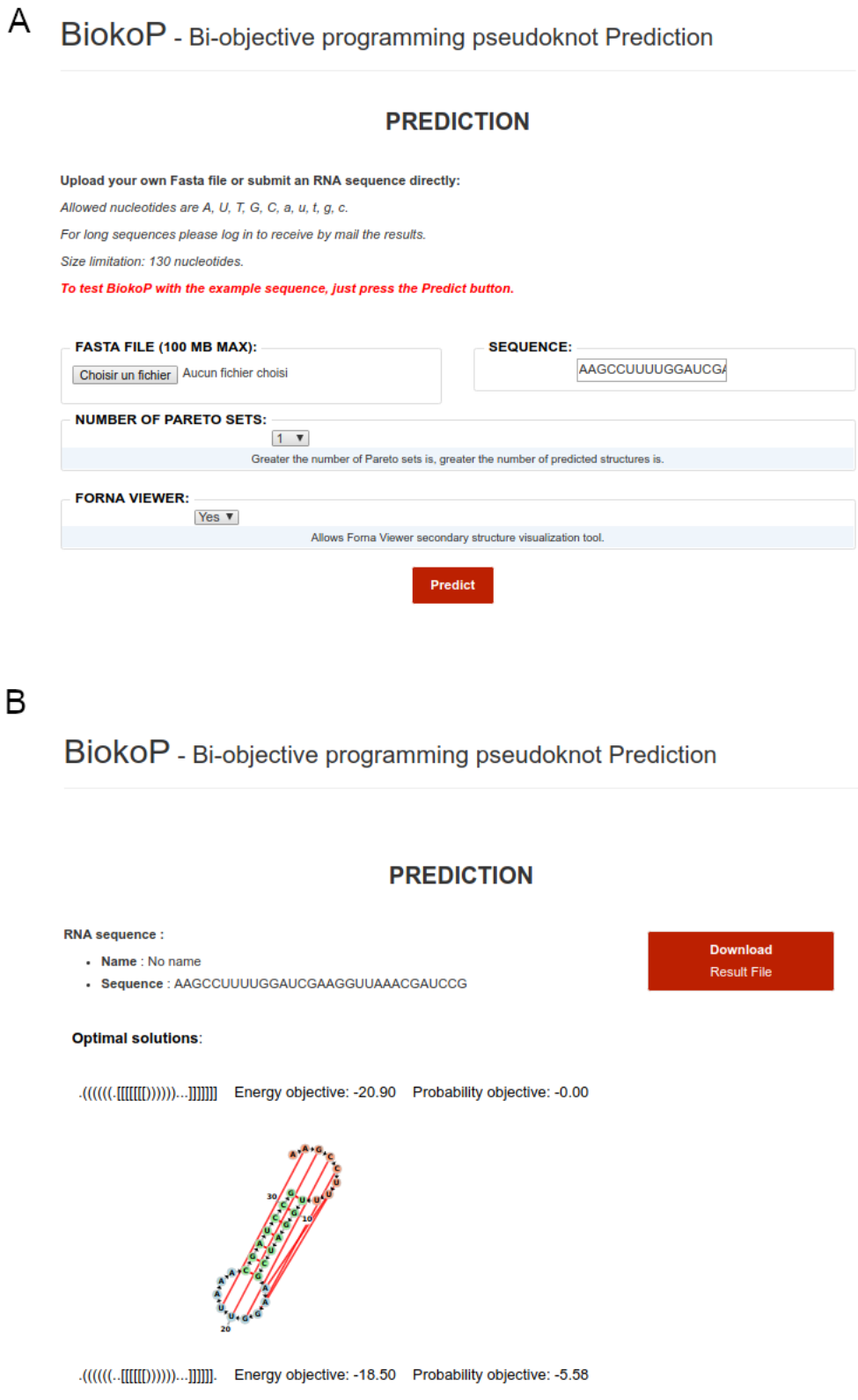

Figure 2.3 : Interface du webserver de l'outil BiokoP. A) Page de formulaire. B) Page des résultats.

$B_{\text{sans nœud}}$, rassemble 145 séquences d'ARN dont les longueurs varient entre 10 et 97 nucléotides.

Les jeux de données $B_{\text{nœud}}$ et $B_{\text{sans nœud}}$ sont disponibles sur la plateforme EvryRNA.

2.3.2 Comparaison des modèles MFE et MEA

Cette section est dédiée à l'étude de deux modèles de prédiction auxquels nous nous référons par Mod1 et Mod2. Pour rappel, Mod1 correspond à la maximisation de la précision attendue et Mod2 correspond à la minimisation de l'énergie libre. Ces deux modèles peuvent générer tout type de pseudonœuds. Mod1^{so} et Mod2^{so} (*so* pour sous-optimal) sont les extensions de ces modèles retournant plusieurs solutions sous-optimales grâce à des contraintes d'interdiction de solutions (voir équations 2.10 et 2.11). Nous avons prédit avec ces deux modèles 30 structures sur le jeu de données $pk168$. Le nombre de structures réelles trouvées par Mod1^{so} et Mod2^{so} est présenté en figure 2.4.

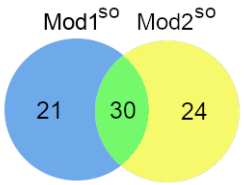

Figure 2.4 : Distribution des structures réelles trouvées par Mod1^{so} et Mod2^{so} sur le jeu de données $pk168$.

Comme nous pouvons le voir, pour 30 ARN, les structures réelles sont trouvées par les deux modèles. Tandis que pour 21 ARN, seul Mod1^{so} trouve les structures réelles, et inversement, pour 24 ARN, seul Mod2^{so} trouve les structures réelles. Cela montre la complémentarité de ces modèles, et bien qu'ayant une base commune, la thermodynamique, ils permettent chacun de trouver des structures réelles différentes.

2.3.3 Évaluation de BiokoP

Dans cette section nous évaluons la capacité de l'outil BiokoP à retourner les structures réelles des ARN. Nous analysons dans quels ensembles de Pareto sont retrouvées ces structures et à quel rang. Nous comparons également BiokoP aux deux modèles combinés Mod1 et Mod2.

Importance de générer plusieurs ensembles de Pareto

Nous étudions ici la distribution des structures réelles trouvées dans les ensembles de Pareto sur le jeu de données $B_{\text{nœud}}$. La structure réelle d'un ARN est la structure correspondant exactement à la structure de référence.

Nous avons étudié la distribution des structures réelles sur 30 structures optimales et sous-optimales. Déterminer le nombre de solutions d'un ensemble de Pareto *a priori* n'est pas possible. Avec les ARN du jeu de données $B_{\text{nœud}}$,

nous avons déterminé qu'il faut en moyenne calculer 5.2 ensembles de Pareto pour obtenir 30 solutions.

La distribution des structures réelles trouvées sont présentées dans le tableau 2.4. Environ la moitié des structures réelles trouvées sont dans le premier ensemble de Pareto (45 sur 83). Ces structures sont Pareto optimales, cela montre la pertinence de combiner les deux modèles thermodynamiques MFE et MEA. Les structures réelles correspondant à des solutions sous-optimales sont distribuées dans les premiers sous-ensembles de Pareto, en particulier dans le deuxième (15) et le troisième (13). Les autres structures sont éparpillées dans les ensembles de Pareto restants.

La position de ces solutions sous-optimales confirme que la structure réelle est souvent une solution sous-optimale dans le modèle thermodynamique. Cela suggère que les solutions sous-optimales prédites par BiokoP sont diversifiées et que notre approche bi-objectif est pertinente pour trouver des solutions sous-optimales. Les premiers ensembles de Pareto sont les plus intéressants, en effet, la qualité des solutions diminue à mesure que le nombre d'ensembles de Pareto augmente. Nous recommandons aux utilisateurs de calculer 3 ensembles de Pareto pour obtenir des solutions qui sont le plus susceptibles d'être des structures réelles.

k^{e} meilleur ensemble de Pareto, $k=$	1	2	3	4	5	6	7	8	9	10	Total
Nombre de structures réelles trouvées	45	15	13	7	1	0	0	0	1	1	83

Tableau 2.4 : Distribution des structures réelles trouvées par BiokoP en fonction du nombre d'ensembles de Pareto générés sur le jeu de données $B_{\text{nœud}}$.

Comparaison de BiokoP avec Mod1$^{\text{so}}$ et Mod2$^{\text{so}}$

Nous étudions ici la distribution des structures réelles trouvées par BiokoP en fonction du nombre de solutions retournées sur le jeu de données $B_{\text{nœud}}$. BiokoP est comparé à Mod1$^{\text{so}}$ et Mod2$^{\text{so}}$ de manière à voir l'apport de la combinaison bi-objectif de ces modèles. Pour rappel, Mod1$^{\text{so}}$ et Mod2$^{\text{so}}$ sont des extensions de Mod1 et Mod2 pouvant générer des solutions sous-optimales. Les résultats sont présentés en figure 2.5.

Les solutions retournées par BiokoP issues d'un même ensemble de Pareto sont par définition non comparables entre elles. Pourtant, afin de comparer les solutions de BiokoP à celles Mod1 et Mod2, nous avons dû ordonner les solutions de BiokoP. Nous avons choisi la procédure suivante pour les ordonner : les solutions d'un ensemble de Pareto sont ordonnées suivant leur distance à la diagonale formée par les deux objectifs, les solutions les plus proches de la diagonale ont un rang inférieur. Cette procédure a été choisi de manière arbitraire pour comparer les solutions de BiokoP à celles des autres outils, mais il existe d'une part, des recherches pour comparer et évaluer des ensembles de Pareto [174], et d'autres part, des recherches sur des méthodes d'aide à la décision pour trier et choisir les "meilleures" solutions [67].

Les résultats montrent que BiokoP prédit plus de structures réelles que Mod1 et Mod2. En effet, ces trois programmes linéaires prédisent la structure

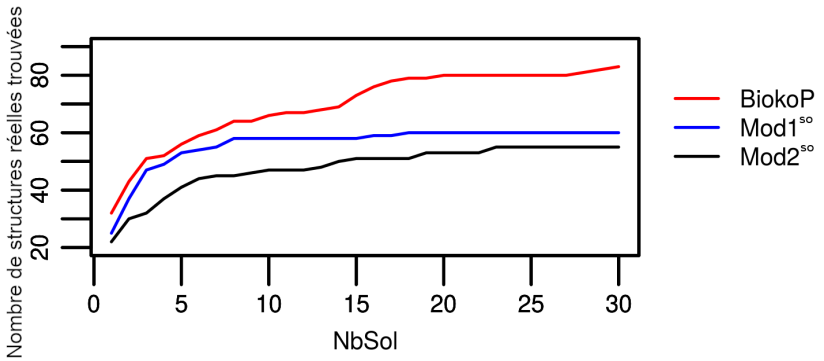

Figure 2.5 : Distribution des structures réelles trouvées par BiokoP, Mod1^{so} et Mod2^{so} en fonction du nombre de solutions retournées (NbSol) sur le jeu de données $B_{\text{nœud}}$.

réelle de respectivement 32, 25 et 23 ARN. Nous avons observé que parmi les structures prédites par Mod1 et Mod2, 12 correspondent aux mêmes ARN. BiokoP trouve toutes les structures réelles prédites par Mod1 et Mod2, ainsi que les structures réelles de 6 ARN pour lesquels Mod1 et Mod2 échouent. Toutes ces structures se trouvent parmi le premier ensemble de Pareto ; c'est-à-dire que toutes les structures prédites par Mod1 et/ou Mod2 sont prédites par BiokoP en tant que solutions optimales. Cela montre que le programme bi-objectif permet de tirer avantage des deux modèles et qu'il permet de trouver de nouvelles solutions.

Plus on génère de solutions avec BiokoP, plus la probabilité de trouver la structure réelle augmente. Cette probabilité augmente rapidement. Nous avons observé qu'après environ 20 solutions retournées par BiokoP (c'est-à-dire environ 2 ou 3 ensembles de Pareto), le nombre de structures réelles trouvées devient stable, ce qui confirme les résultats de la section précédente.

Quant à Mod1^{so} et Mod2^{so}, le nombre de structures réelles trouvées atteint rapidement un plateau à partir de 7 à 8 solutions retournées. Cela est dû au manque de diversité des solutions retournées. En effet, les solutions sous-optimales sont pour la plupart similaires à la solution optimale : elles sont dérivées de la solution optimale par l'ajout ou la suppression de quelques appariements. Quand la structure réelle est proche de la solution optimale, elle peut être trouvée rapidement dans les solutions sous-optimales, ce qui explique l'augmentation rapide du nombre de structures réelles trouvées pour un petit nombre de solutions retournées. En revanche, lorsque la structure réelle est éloignée de la solution optimale, celle-ci peut ne jamais être trouvée ou très tardivement.

D'un point de vue plus général, cette expérience montre que la probabilité de prédiction de la structure réelle est plus importante parmi les solutions optimales et sous-optimales retournées par BiokoP que parmi celles retournées par Mod1^{so} et Mod2^{so}.

2.3.4 Comparaison de BiokoP avec la littérature

Outils considérés

Pour évaluer les performances de BiokoP, nous l'avons comparé avec des outils de la littérature capables de prédire des structures secondaires d'ARN avec pseudonœuds et pouvant retourner plusieurs solutions. Quatre outils sont disponibles dans la littérature, à savoir pKiss [101], McGenus [30], ainsi que MC-Fold [164] et MC-Flashfold [46]. Nous avons également comparé BiokoP avec IPknot [194] et RNAsubopt [130]. IPknot retourne une structure secondaire pouvant contenir des pseudonœuds, tandis que RNAsubopt retourne plusieurs structures secondaires mais sans pseudonœuds. Tous ces outils sont décrits en section 2.1.2.

Lors de l'évaluation, nous avons comparé les 30 premières solutions retournées par BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold et RNAsubopt et la solution retournée par IPknot. pKiss (version 2.2.12) a été exécuté avec les paramètres par défaut. Nous avons utilisé l'option -relativeDeviation pour obtenir 30 solutions pour chaque ARN. McGenus (version 7.0) a été exécuté également avec les paramètres par défaut, avec l'option -nsuboptimal pour obtenir 30 solutions. MC-Fold et MC-Flashfold ont été utilisés avec l'option -ft pour obtenir 30 solutions. Nous avons exécuté RNAsubopt (version 2.3.3) avec l'option -e pour obtenir 30 solutions et l'option -s pour trier les solutions selon l'énergie. IPknot (version 0.0.4) a été exécuté avec les paramètres énergétiques de Dirks et Pierce [62] et avec les options -g 2 et -g 4.

Les solutions obtenues par pKiss, McGenus, MC-Fold, MC-Flashfold et RNAsubopt sont ordonnées dans l'ordre croissant des énergies. Les solutions de BiokoP appartenant au même ensemble de Pareto sont retournées dans un ordre arbitraire et ne sont pas comparables ; nous les avons alors triées selon la diagonale, comme expliqué en section précédente. Les meilleures solutions sont celles optimisant de manière égale les deux objectifs, c'est-à-dire les solutions les plus proches de la diagonale.

Résultats globaux

Les résultats obtenus par les outils BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold et RNAsubopt, sur le jeu de données $B_{\text{nœud}}$, sont présentés dans les tableaux 2.5 et 2.6. Les tableaux 2.5 et 2.6 rassemblent les moyennes pondérées des sensitivités et PPV en fonction du nombre de solutions retournées par chaque outil.

Nous pouvons voir que BiokoP obtient de meilleures sensitivités que tous les autres outils et que lorsque le nombre de solutions augmente, l'écart entre la sensitivité de BiokoP et celle des autres outils augmente également. Par ailleurs, BiokoP obtient de meilleurs PPV comparé à McGenus, MC-Fold, MC-Flashfold et RNAsubopt et ses PPV sont comparables à ceux de pKiss.

La figure 2.6 présente les moyennes pondérées des F_1-scores obtenues par chaque outil, en fonction du nombre de solutions retournées. Comme nous pouvons le voir, BiokoP obtient des F_1-scores plus élevés que tous les autres outils. Les F_1-scores de BiokoP, ainsi que ceux de MC-Fold, MC-Flashfold et

NbSol	Sensitivité écart-type											
	BiokoP		pKiss		McGenus		MC-Fold		MC-Flashfold		RNAsubopt	
1	**80.6**	22.3	79.5	24.2	73.4	26.6	58.2	38.7	31.5	29.5	53.1	23.3
2	**80.6**	22.3	77.6	23.5	67.1	26.7	58.0	38.5	31.8	29.4	52.0	23.1
3	**80.1**	22.4	75.8	23.6	63.8	30.5	57.9	38.4	32.1	29.5	51.4	22.9
4	**79.5**	22.8	74.4	23.8	62.2	30.8	57.7	38.1	32.3	29.5	50.8	22.9
5	**79.0**	23.1	73.1	24.0	61.0	31.0	57.5	38.0	32.5	29.5	50.4	22.9
10	**77.0**	23.6	68.8	24.1	57.4	31.9	56.9	37.7	32.8	29.5	48.7	23.2
15	**75.8**	23.6	65.7	24.5	55.7	32.6	55.8	37.3	33.0	29.4	48.0	23.4
20	**75.1**	23.6	63.2	25.0	54.2	33.1	54.9	37.2	33.1	29.3	47.1	23.6
25	**74.5**	23.5	61.2	25.3	53.3	33.4	54.4	37.0	33.2	29.2	46.4	23.9
30	**73.8**	23.5	59.5	25.5	53.3	33.4	53.7	37.0	33.2	29.1	45.8	24.1

Tableau 2.5 : Moyennes pondérées en fonction du nombre de solutions (NbSol) des sensitivités obtenues par BiokoP, pKiss, McGenus, MC-Fold et MC-Flashfold sur le jeu de données $B_{\text{nœud}}$.

NbSol	PPV écart-type											
	BiokoP		pKiss		McGenus		MC-Fold		MC-Flashfold		RNAsubopt	
1	75.0	25.5	**75.1**	26.6	74.1	28.6	44.6	30.6	25.8	26.1	67.7	32.7
2	73.2	25.5	**74.8**	26.2	69.7	31.6	44.5	30.6	25.9	26.1	66.8	32.6
3	71.6	25.7	**74.2**	26.4	67.3	32.7	44.7	30.7	26.3	26.3	66.5	32.3
4	70.5	26.0	**73.5**	26.6	66.2	33.3	44.7	30.6	26.6	26.5	65.9	32.3
5	69.7	26.2	**72.9**	26.8	65.5	33.6	44.7	30.6	26.7	26.6	65.5	32.3
10	67.4	26.4	**70.9**	27.1	62.6	35.3	44.5	30.5	27.1	26.8	63.8	32.3
15	66.4	26.4	**69.3**	27.6	61.0	36.1	44.0	30.3	27.5	26.8	62.6	32.4
20	65.9	26.4	**67.9**	28.1	59.4	36.8	43.5	30.3	27.6	26.7	61.5	32.6
25	65.5	26.3	**66.7**	28.5	58.4	37.2	43.0	30.2	27.6	26.6	60.6	32.8
30	65.0	26.3	**65.2**	28.9	58.5	37.2	42.5	30.2	27.6	26.5	59.7	32.9

Tableau 2.6 : Moyennes pondérées en fonction du nombre de solutions (NbSol) des PPV obtenus par BiokoP, pKiss, McGenus, MC-Fold et MC-Flashfold sur le jeu de données $B_{\text{nœud}}$.

RNAsubopt sont plutôt stables. En effet, il y a seulement une baisse de 10 points pour BiokoP quand le nombre de solutions varie entre entre 1 et 30, alors qu'il y a une baisse de 15 et 18 points pour pKiss et McGenus respectivement. Cela suggère que la qualité des structures prédites par BiokoP, MC-Fold, MC-Flashfold et RNAsubopt est stable même quand la quantité des solutions retournées augmente.

Lorsqu'une seule solution est retournée, BiokoP obtient des résultats similaires à IPknot : IPknot obtient une sensitivité moyenne de 80.6% (80.6% pour BiokoP), un PPV moyen de 75.1% (75.0% respectivement) et un F_1-score moyen de 77.0% (77.0% respectivement).

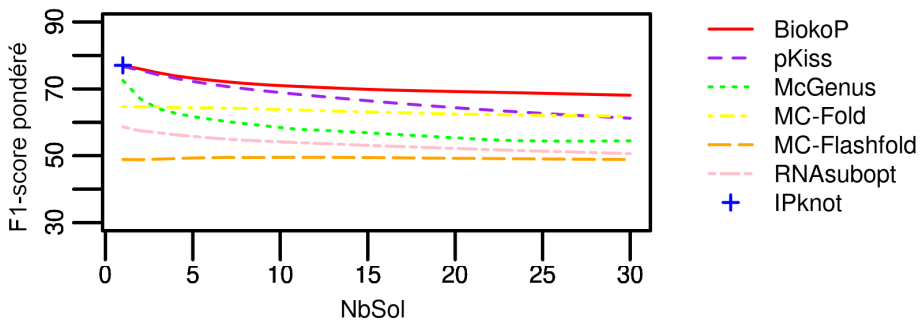

Figure 2.6 : F_1-scores moyens pondérés obtenus sur le jeu de données $B_{\text{nœud}}$ par BiokoP, pKiss, McGenus et IPknot, en fonction du nombre de solutions (NbSol).

Résultats concernant les solutions optimales

Contrairement à tous les autres outils de la littérature, qui ne prédisent qu'une seule structure optimale, BiokoP en prédit plusieurs (le 1$^{\text{er}}$ ensemble de Pareto). Il n'est pas évident de comparer la solution optimale donnée par pKiss, McGenus, MC-Fold, MC-Flashfold, RNAsubopt ou IPknot à l'ensemble des solutions optimales données par BiokoP. Les solutions appartenant à un ensemble de Pareto ne sont pas comparables entre elles. La figure 2.7 présente les F_1-scores des solutions optimales obtenues par BiokoP, pKiss, McGenus, IPknot, MC-Fold, MC-Flashfold et RNAsubopt pour chaque ARN du jeu de données $B_{\text{nœud}}$. Les ARN sont triés dans l'ordre croissant des F_1-scores maximums obtenus par BiokoP. Pour BiokoP, les F_1-scores présentés sont le minimum et le maximum obtenus pour chaque ARN parmi l'ensemble de Pareto optimal. Les résultats montrent que BiokoP obtient de meilleures solutions que pKiss pour 84 ARN, que McGenus pour 103 ARN, que MC-Fold pour 153 ARN, que MC-Flashfold pour 112 ARN et que RNAsubopt pour 156 ARN (sur un total de 198). À l'inverse, pour un certain nombre d'ARN, ces outils obtiennent de meilleures solutions que BiokoP (pKiss : 54 ; McGenus : 39 ; MC-Fold : 44 ; MC-Flashfold : 1 ; RNAsubopt : 37). Les résultats montrent également que BiokoP obtient 61 meilleures solutions comparé à IPknot, alors qu'IPknot ne retourne aucune meilleure solution comparé à BiokoP.

Le fait de retourner un ensemble de solutions optimales permet à BiokoP d'obtenir de meilleurs résultats. Cependant, nous pouvons voir que l'écart de performance des solutions obtenues par BiokoP peut être important (l'écart entre le minimum et le maximum).

Résultats selon les types de pseudonœuds

Le programme linéaire de BiokoP peut prédire tous les types de pseudonœuds, ce qui n'est pas forcément le cas des outils de la littérature. Nous étudions ici les résultats de BiokoP, ainsi que des outils pKiss, McGenus, IPknot, MC-Fold, MC-Flashfold et RNAsubopt sur chaque type de pseudonœuds présent dans le jeu de données $B_{\text{nœud}}$.

Les F_1-scores obtenus par les différents outils, en fonction du nombre de

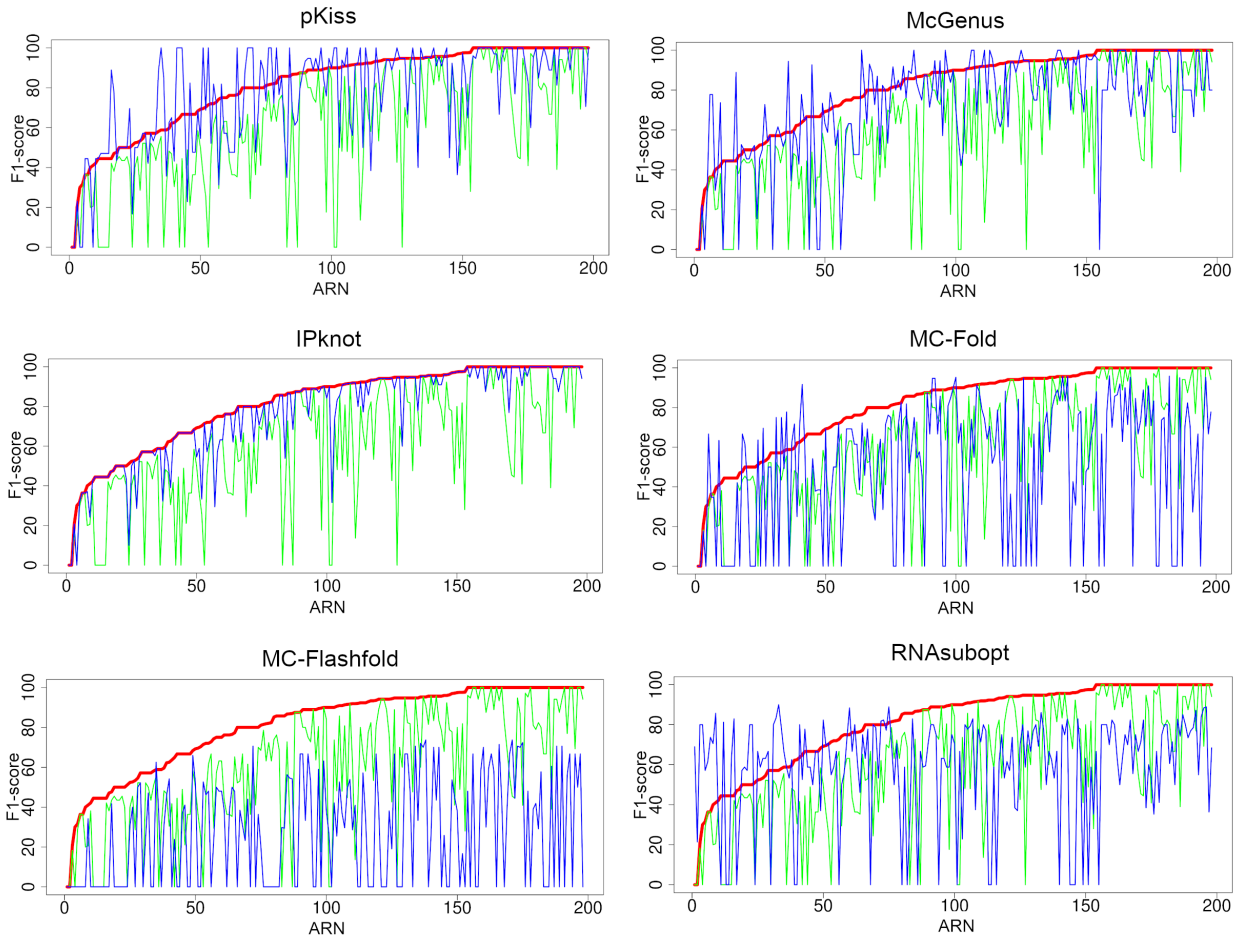

Figure 2.7 : F_1-scores des solutions optimales obtenues par BiokoP comparé à pKiss, McGenus, IPknot, MC-Fold, MC-Flashfold et RNAsubopt sur le jeu de données $B_{\text{nœud}}$. Les F_1-scores de BiokoP, minimum et maximum, sont respectivement en vert et en rouge, et celui des autres outils est en bleu.

solutions retournées et du type de pseudonœuds, sont présentés en figure 2.8. Les résultats pour les pseudonœuds de type H sont très similaires aux résultats du jeu de données $B_{\text{nœud}}$ entier. En effet, les pseudonœuds de type H sont très largement représentés dans ce jeu de données (154 sur 198 ARN). Les pseudonœuds de type HHH et HLout sont mieux prédits par McGenus (F_1-scores moyens à 84% et 73% respectivement). Cependant, BiokoP obtient de meilleurs résultats que tous les autres outils pour les pseudonœuds de type HLin (70%) et LL (75%). Comparé à IPknot, BiokoP obtient de meilleurs résultats pour les types HHH et LL, et des résultats similaires pour les autres types.

Le programme linéaire de BiokoP a été conçu de manière à pouvoir prédire tous les types de pseudonœuds. Les résultats homogènes montrent qu'effectivement, BiokoP prédit avec une précision similaire les différents types de pseudonœuds. Les résultats de BiokoP ne sont jamais inférieurs à 70% quelque soit le type de pseudonœuds et quelque soit le nombre de solutions retournées. Ce n'est pas le cas des autres outils, pour lesquels les résultats varient selon le

type de pseudonœuds. Néanmoins, comme la répartition des pseudonœuds de différents types dans le jeu de données est très inégales (3 HHH, 26 HLout, 4 HLin, 11 LL et 154 H), des études ultérieures avec des données supplémentaires doivent être menées pour confirmer ces résultats.

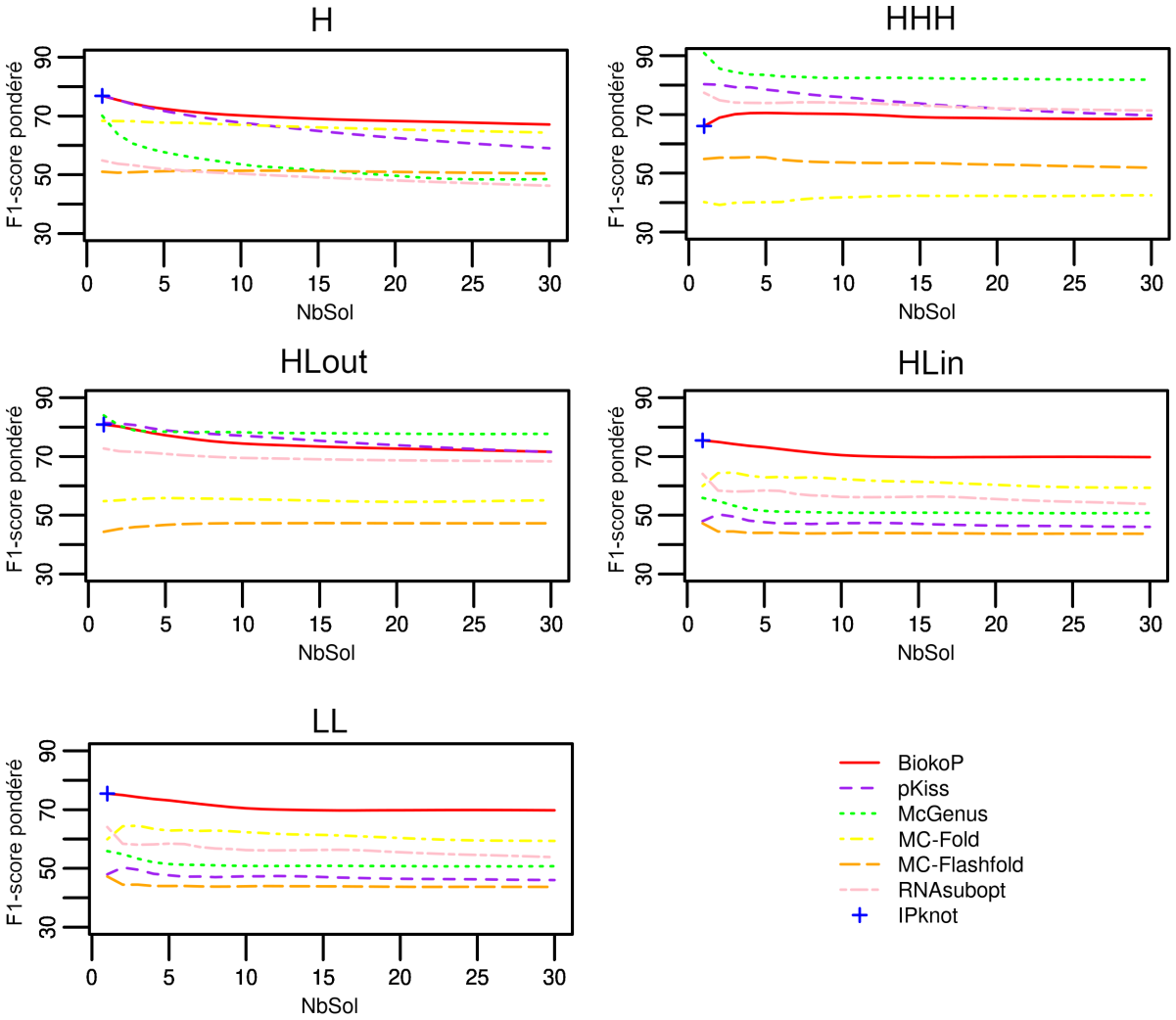

Figure 2.8 : F_1-scores moyens pondérés de BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold et IPknot, en fonction du type de pseudonœuds et du nombre de solutions retournées (NbSol). Tests effectués sur le jeu de données $B_{\text{nœud}}$ (3 HHH, 26 HLout, 4 HLin, 11 LL et 154 H).

Résultats sur des ARN sans pseudonœud

Afin de déterminer la capacité de BiokoP à prédire des structures sans pseudonœuds, nous l'avons évalué sur le jeu de données $B_{\text{sans nœud}}$, qui ne contient que des ARN sans pseudonœuds, et comparé aux outils pKiss, McGenus, IPknot, MC-Fold, MC-Flashfold et RNAsubopt.

Dans la figure 2.9 sont présentés les F_1-scores moyens pondérés obtenus en fonction du nombre de solutions retournées. BiokoP, pKiss et RNAsubopt obtiennent des résultats comparables, ce qui montre que BiokoP, tout comme

pKiss, sont capables de prédire des structures sans pseudonœuds avec une précision similaire à des structures avec pseudonœuds. Nous nous attendions à une plus grande différence avec RNAsubopt car cet outil a été développé pour générer des structures sans pseudonœuds. Ces résultats suggèrent que BiokoP et pKiss ne subissent pas de biais de par leur modélisation. En revanche, nous pouvons voir que les outils MC-Fold et MC-Flashfold obtiennent de meilleurs résultats que les autres outils sur ce jeu de données.

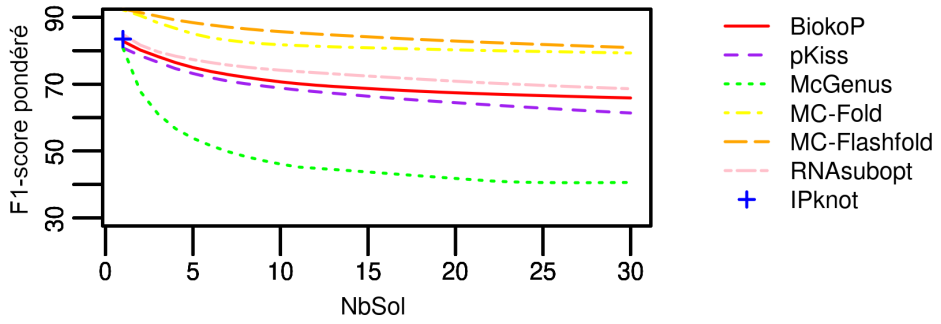

Figure 2.9 : F_1-scores moyens pondérés obtenus par BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold, IPknot et RNAsubopt sur le jeu de données $B_{\text{sans nœud}}$, en fonction du nombre de solutions (NbSol).

Cas d'étude

Nous avons testé BiokoP sur deux ARN dont la structure secondaire est bien connue, un ARN avec pseudonœud et un ARN sans pseudonœud. Nous avons choisi le pseudonœud PK4 (type H) de l'ARN télomérase de Legionella Pneumophila[236] et l'ARN sans pseudonœud en forme de marteau [82]. Pour cela nous avons évalué les 30 premières solutions retournées par BiokoP et les autres outils.

La structure de référence du pseudonœud PK4 et la meilleure solution (selon le F_1-score) retournée par BiokoP, pKiss, McGenus, RNAsubopt, IPknot, MC-Fold et MC-Flashfold, sont montrées en figure 2.10 A. La visualisation des structures a été obtenue grâce à l'outil forna [109] disponible à partir du ViennaRNA Web Service. BiokoP trouve la structure de référence de cet ARN au rang 15, dans le premier ensemble de Pareto sous-optimal. McGenus, pKiss, RNAsubopt, MC-Fold et MC-Flashfold trouvent la structure globale mais n'arrivent pas à trouver le pseudonœud (en partie ou complètement). IPknot prédit la structure de référence à un appariement près de différence. Tous les outils obtiennent un F_1-score élevé pour cet ARN, mais BiokoP et IPknot sont les seuls à prédire le pseudonœud de manière exacte.

Nous avons ensuite étudié l'ARN sans pseudonœud en forme de marteau [82] (figure 2.10 B). Cette structure particulière est composée de trois hélices. Elle est très étudiée pour comprendre la relation structure-fonction des ARN. La meilleure solution retournée par BiokoP a un F_1-score de 81.3% et est retournée au rang 9 dans le premier ensemble de Pareto sous-optimal. BiokoP trouve les deux premières hélices et quelques appariements de la troisième, mais prédit un pseudonœud de deux appariements. pKiss, McGenus, et MC-Fold trouvent

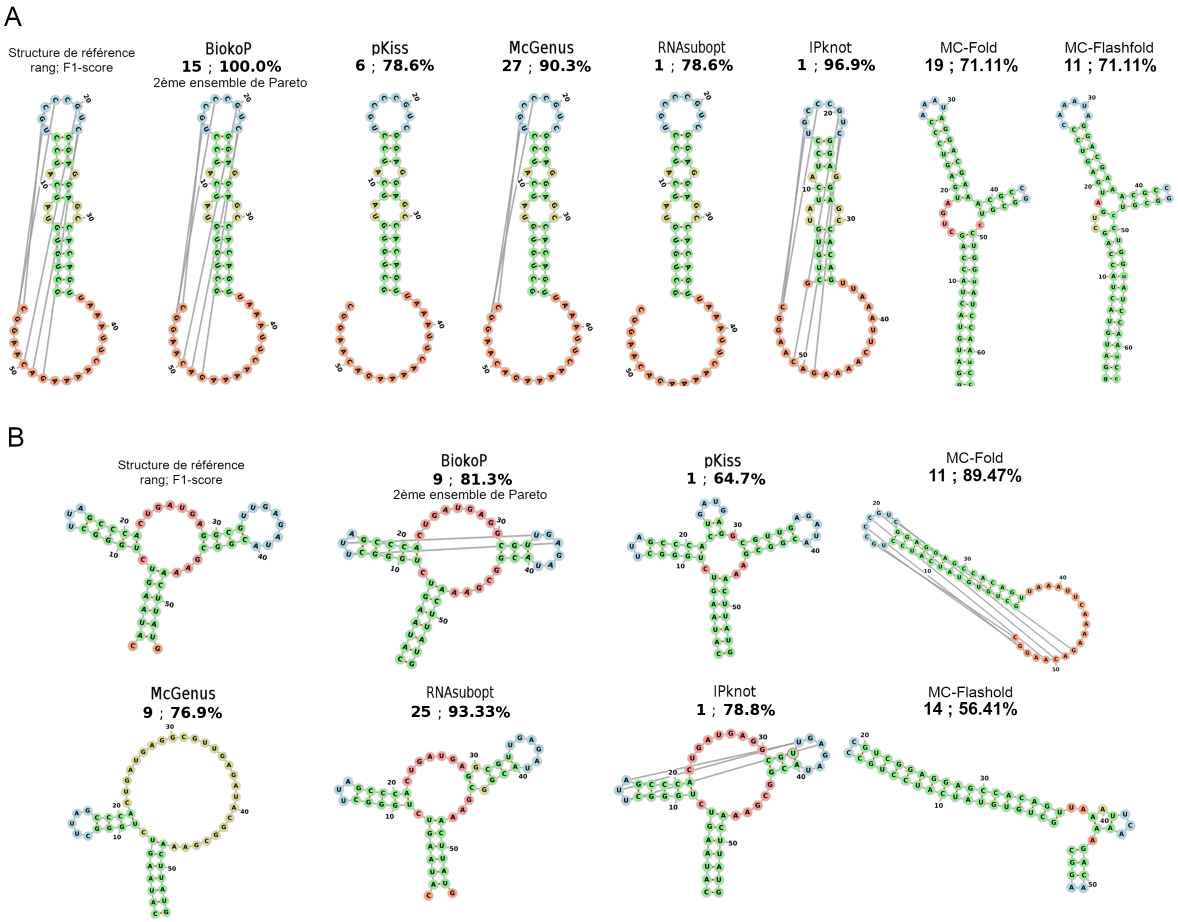

Figure 2.10 : Prédictions de structures secondaires avec BiokoP, pKiss, McGenus, MC-Fold, MC-Flashfold, IPknot et RNAsubopt pour le pseudonœud (type H) PK4 de l'ARN télomérase [236] (A) et un pseudonœud en forme de marteau [82] (B). Pour chaque outil est donné la meilleure structure prédite parmi 30 solutions retournées. Les visualisations ont été obtenues à l'aide de l'outil forna disponible à partir du ViennaRNA Web Service [109].

également les deux premières hélices. Cependant, pKiss prédit une boucle en épingle à cheveux qui n'existe pas dans la structure de référence et McGenus ne trouve aucun appariement de la troisième hélice. RNAsubopt trouve les deux premières hélices et quelques appariements de la troisième. MC-Flashfold trouve une partie des bons appariements. La structure prédite par IPknot est identique à celle prédite par BiokoP à la différence que le pseudonœud a un appariement supplémentaire. RNAsubopt obtient de meilleurs résultats, cependant, sa structure n'est trouvée qu'au 25^{e} rang.

2.3.5 Temps d'exécution

La complexité en temps de résolution d'un programme linéaire en nombres entiers est exponentielle et est fonction du nombre de variables, qui dépend dans notre cas largement de la longueur de la séquence de l'ARN. De plus, notre programme entier est ici bi-objectif, ce qui signifie que le programme

entier mono-objectif associé doit être résolu de nombreuses fois. Pour un ARN de 30 nucléotides, en considérant une prédiction de 30 solutions, BiokoP prend 22 secondes à s'exécuter (OS Debian, 24 Intel Xeon CPU X5660 2.8 GHz, 64Go RAM). Pour le pseudonœud PK4 (type H) de l'ARN télomérase de Legionella Pneumophila[236] de 55 nucléotides, BiokoP a pris 5 minutes et 30 secondes. Pour les plus grands ARN que nous avons testé (environ 150 nucléotides), le programme est exécuté en quelques heures maximum.

Les autres outils sont exécutés rapidement, en quelques secondes, hormis MC-Fold qui peut parfois mettre environ 1 heure (mais dont la version MC-Flashfold est rapide), et IPknot qui peut mettre quelques minutes à s'exécuter. Nous travaillons à améliorer le temps d'exécution de BiokoP, des pistes sont explorées dans le chapitre 5.

2.4 Discussion

Nous avons développé un algorithme générique original pour résoudre les programmes linéaires en nombres entiers bi-objectif qui permet de générer toutes les solutions optimales (l'ensemble de Pareto exact), ainsi que des solutions sous-optimales (les k sous-ensembles de Pareto exacts). Nous avons proposé un programme linéaire en nombres entiers bi-objectif qui combine deux modèles de prédiction de structures secondaires d'ARN avec pseudonœuds, à savoir MEA et MFE. L'implémentation de cet algorithme avec ce programme linéaire bi-objectif a permis le développement de l'outil BiokoP (*Bi-objective programming pseudoknot Prediction*).

Comme notre méthode est basée sur l'optimisation multi-objectif, BiokoP retourne plusieurs ensembles de solutions (les k meilleurs ensembles de Pareto). Ils sont retournés selon l'ordre d'optimalité. Cependant, toutes les solutions appartenant à un même ensemble de Pareto ne sont pas comparables et ne peuvent donc être ordonnées à l'intérieur même d'un ensemble tandis que les autres outils de la littérature retournent des solutions triées de manière optimale. C'est pourquoi la comparaison des résultats de BiokoP avec les autres outils de la littérature a soulevé des difficultés. En effet, en optimisation multi-objectif, définir des mesures de performance pour les ensembles de Pareto est un sujet de recherche à part entière [40]. Nous avons choisi de trier les solutions en supposant qu'une solution était meilleure si elle optimisait également les deux objectifs, c'est-à-dire si la solution était proche de la diagonale formée par les deux objectifs. Ce tri nous a permis de montrer les bénéfices de la combinaison des deux modèles de prédiction Mod1 et Mod2. En effet, les tests effectués ont montré que la première solution retournée par BiokoP avait plus de chance d'être la structure de référence d'un ARN que la première solution retournée par Mod1 et Mod2 (figure 2.5). Les tests ont également montré que cette combinaison permet d'obtenir des résultats homogènes par rapport au type de pseudonœuds, au nombre de solutions retournées et à la présence ou non de pseudonœuds, comparé aux autres outils de la littérature (figures 2.6, 2.8 et 2.9).

Cependant, ce tri montre des limites. Elles sont illustrées par les résultats montrant le détail des types de pseudonœuds, dans lesquels nous pouvons

voir que la meilleure solution n'est pas toujours la première selon ce tri (figure 2.8, type HHH).

Afin d'avoir une idée plus précise de la qualité globale des solutions du premier ensemble de Pareto, nous avons comparé ces solutions aux solutions optimales des autres outils en étudiant la meilleure et la pire des solutions de l'ensemble de Pareto (selon le F_1-score). Les résultats ont montré que dans la plupart des cas, BiokoP trouve plus souvent de meilleures solutions que les autres outils (figure 2.7).

Pour résumer, les tests que nous avons menés ont montré que :

— La génération d'ensembles de Pareto sous-optimaux est pertinente, en effet, ils peuvent contenir des structures réelles.

— La combinaison de Mod1 et Mod2 est également pertinente : BiokoP retourne toutes les structures réelles trouvées par Mod1 et Mod2 et trouve de nouvelles structures.

— BiokoP, comparé aux outils de la littérature, est le meilleur compromis entre le nombre de fois où la structure réelle est trouvée et la qualité des prédictions.

3 Prédiction de structures secondaires de complexes d'ARN

Les ARN peuvent se lier et former des complexes ayant des fonctions catalytiques. C'est par exemple le cas du ribosome, composé des ARN 5S, 5,8S, 18S et 28S chez les eucaryotes, qui est responsable de la traduction des ARN messagers en protéines (voir section 1.1.5, page 13). Le ribosome est également composé de protéines, mais ce sont les ARN qui sont responsables de l'activité catalytique du complexe. Il existe également des complexes composés uniquement d'ARN comme l'hexamère cyclique du bactériophage φ29, servant à infecter les bactéries.

Les complexes d'ARN sont formés par des interactions canoniques et non-canoniques. La prédiction bioinformatique de structures est surtout consacrée aux structures d'ARN, ainsi qu'aux interactions ARN-ARN, mais très peu en comparaison aux complexes d'ARN. En effet, dans la littérature, seuls trois outils sont disponibles pour la prédiction de structures de complexes d'ARN, MultiRNAfold [10], NUPACK [248] et Nanofolder [23].

Comme nous l'avons vu au chapitre précédent, les structures sous-optimales sont importantes car les ARN peuvent avoir plusieurs structures selon leur environnement et la structure réelle ne correspond pas toujours à la structure d'énergie minimum. Cela s'applique également à la prédiction de structures secondaires de complexes d'ARN. Pourtant, parmi les outils disponibles, seul NUPACK peut retourner des structures sous-optimales.

Nous avons également étudié les pseudonœuds au chapitre précédent. Leur prédiction n'est pas toujours possible par les outils disponibles pour les ARN seuls, et c'est également le cas pour les complexes d'ARN. Seul Nanofolder peut prédire des pseudonœuds. Dans les complexes d'ARN, on distingue un nouveau type de pseudonœuds, présent dans les interactions ARN-ARN, ce sont les pseudonœuds dits "externes", en opposition aux pseudonœuds dans les ARN seuls qui sont dits "internes" (voir figure 3.1). Nanofolder peut prédire ces deux types de pseudonœuds.

Parmi les outils disponibles de la littérature, il n'en existe aucun qui soit capable de générer des structures secondaires de complexes d'ARN ayant des pseudonœuds et pouvant retourner des structures sous-optimales. Nous proposons ici une nouvelle méthode de prédiction tirant parti des nombreux outils dédiés à la prédiction de structures secondaires d'ARN seuls et d'interaction

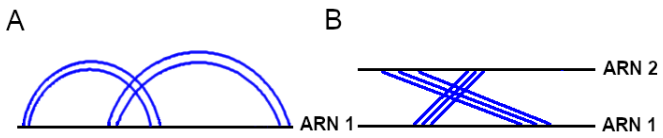

Figure 3.1 : Définitions des motifs d'un complexe d'ARN. A) Structure secondaire avec pseudonœud. B) Interaction ARN-ARN avec pseudonœud externe.

ARN-ARN. En effet, nous assimilons une structure de complexe d'ARN à un ensemble de structures secondaires d'ARN et d'interactions ARN-ARN. Notre méthode prend en entrée un ensemble de séquences d'ARN d'intérêts, plusieurs structures secondaires par ARN et plusieurs interactions ARN-ARN par paire d'ARN, générées par des outils de prédiction. Notre méthode vise à trouver les meilleures combinaisons possibles des entrées formant des structures de complexe d'ARN.

Notre méthode est basée sur un problème connu en théorie des graphes, le problème de la clique maximum. Le problème de la clique maximum est NP-difficile et il existe de nombreuses heuristiques pour le résoudre de manière approchée en temps polynomial. Nous avons modifié une heuristique pour pouvoir récupérer un ensemble de cliques et non pas une seule.

Cela nous permet d'aborder, dans un premier temps, une méthode approchée en optimisation mono-objectif (présentée dans ce chapitre) en vue de proposer un outil multi-objectif pour prédire des structures secondaires de complexes d'ARN, intégrant des données structurales et des contraintes utilisateurs. La méthode multi-objectif fera l'objet du chapitre suivant.

Nous avons ainsi développé l'outil RCPred (*RNA Complex Prediction*) à partir de cette heuristique. RCPred peut prédire des structures secondaires de complexes d'ARN avec pseudonœuds et des structures sous-optimales. Il est basé sur le modèle de minimisation de l'énergie libre (MFE).

Ce chapitre débute avec un état de l'art des outils de prédiction de la structure secondaire d'un complexe d'ARN. Puis, nous proposons deux nouvelles méthodes pour prédire ces structures avec la programmation linéaire. Nous décrivons ensuite comment résoudre le problème de la clique maximum avec l'heuristique appelée *Breakout Local Search* et comment nous avons adapté cette heuristique à notre problème. Enfin, nous présentons notre nouvel outil, RCPred, de prédiction de structures secondaires de complexes d'ARN. Enfin, la dernière section de ce chapitre est consacrée à l'évaluation de RCPred.

3.1 État de l'art

Cet état de l'art est consacré aux méthodes et outils de prédiction d'interaction ARN-ARN et de structures secondaires de complexes d'ARN.

Nous présentons les méthodes et outils de prédiction de structures secondaires de complexes d'ARN en fonction de leur capacité à prédire des pseudonœuds et/ou des structures sous-optimales.

3.1.1 Prédiction d'interaction ARN-ARN

De nombreux outils existent pour prédire la structure secondaire de d'interaction ARN-ARN. Les premiers outils prédisent des interactions ARN-ARN du type ciblé/ciblant, où un ARN est la cible (dans la plupart des cas un petit ARN) d'un autre. Ils utilisent le plus souvent le modèle de minimisation de l'énergie libre pour prédire uniquement les appariements inter-acides nucléiques. Parfois le modèle est simplifié : par exemple, dans l'outil TargetRNA [215], qui permet la prédiction d'interactions entre un petit ARN et un ARNm, l'énergie est calculée grâce au calcul d'un score d'hybridation pour chaque appariement. Certains prennent également en compte la concentration des molécules dans les échantillons (OligoWalk [138]), ce qui permet d'avoir une prédiction plus réaliste quand à la probabilité d'interaction des deux ARN.

Un outil, appelé RNAhybrid [117], prédisant des interactions entre des microARN et des ARNm, est capable de prédire plusieurs sites d'interaction ainsi que des interactions sous-optimales. Dans l'ensemble logiciel ViennaRNA [130], un outil, appelé RNAduplex, est très similaire à RNAhybrid et permet également d'obtenir des interactions sous-optimales.

Tous les outils présentés précédemment utilisent uniquement les appariements inter-ARN (entre deux ARN) pour la prédiction des interactions, d'autres outils utilisent également les appariements intra-ARN (internes à chaque ARN). Ces derniers utilisent les probabilités d'accessibilités et peuvent éventuellement prédire la structure jointe.

Un outil de ViennaRNA, appelé RNAplex [209], permet de prédire des interactions entre un petit ARN et un ARNm, dont des interactions sous-optimales. Il ne tient pas compte des appariements intra-ARN. Cet outil est très rapide car il est destiné à être appliqué sur des génomes entiers. Pour être rapide, il utilise un modèle thermodynamique simplifié basé sur celui de RNAhybrid et un algorithme de repliement équivalent à celui de Zuker et Stiegler [259], où seules les boucles internes sont autorisées. Son amélioration, appelée RNAplex-a [210], peut estimer l'accessibilité si un profil d'accessibilité de l'outil RNAplfold (ViennaRNA) lui est fourni.

Un outil, appelé intaRNA [31], est dédié à la prédiction d'interaction de petits ARN avec un ARNm. Pour prédire ces interactions, il calcule l'accessibilité de chaque nucléotide et utilise comme score la somme des énergies d'hybridation et des énergies d'accessibilité.

Pour prédire des interactions d'ARN longs, un outil, appelé RIblast [76] a été développé. Il utilise le même modèle thermodynamique que RNAplex et intaRNA et utilise un algorithme plus rapide basé sur la méthode *seed and extension* qui détecte les régions potentielles d'hybridation à partir d'un tableau de suffixes élaborés grâce aux séquences.

Un récapitulatif des outils est donné dans le tableau 3.1.

3.1.2 Prédiction de complexes d'ARN

La majorité des outils de prédiction de complexes d'ARN sont capables de prédire uniquement la structure jointe d'un duplexe d'ARN (prédiction du site d'interaction et des appariements internes de chaque ARN). Nous nous

Référence	Modèle	Outil	Accessibilité	Structures sous-optimales
Mathews et al., 1999 [138]	MFE	OligoWalk		
Krüger et Rehmsmeier, 2006 [117]	MFE	RNAhybrid		✓
Tjaden, 2008 [215]	MFE	TargetRNA		
Busch et al., 2008 [31]	MFE	IntaRNA	✓	✓
Tafer et Hofacker, 2008 [209]	MFE	RNAplex		✓
Lorenz et al., 2011 [130]	MFE	RNAduplex		✓
Fukunaga et Hamada, 2016 [76]	MFE	RIblast	✓	✓
Kato et al., 2017 [108]	MEA	RactIPAce	✓	

Tableau 3.1 : Résumé de l'état de l'art des outils de prédiction de structures secondaires d'un duplexe d'ARN.

intéressons ici aux méthodes dédiées à la prédiction de structures de complexes d'ARN composés de plus de deux ARN. Cependant, nous pouvons néanmoins citer en exemple des outils dédiés à la structures des duplexes [245, 151, 191, 130, 168, 4, 6]. La plupart de ces outils sont basés sur le modèle MFE et utilise la programmation dynamique. Un outil, appelé RactIPAce [108], utilise le modèle MEA et les probabilités d'accessibilité avec la programmation linéaire pour prédire les interactions ARN-ARN.

Beaucoup moins d'outils de prédiction de complexes d'ARN, composés de plus de deux ARN, ont été proposés. Le premier outil, appelé MultiRNAfold [10], a été développé par Andronescu et al. en 2005. Dans cet outil, les ARN d'intérêts sont liés tous ensemble bout à bout par des boucles spéciales de manière à former un long ARN. Par la suite, la structure d'énergie minimale est calculée grâce à un algorithme de programmation dynamique inspiré par l'algorithme de Zuker et Stiegler [259]. Cet outil permet de calculer une seule structure secondaire et n'inclut pas la prédiction de pseudonœuds.

L'ensemble logiciel NUPACK [248], rassemblant divers outils dédiés aux ARN, inclut un outil de prédiction de structures secondaires de complexes d'ARN. Dans cet outil, le calcul de la fonction de partition est étendu à n ARN, ainsi que le calcul de la structure d'énergie minimale. Cet outil permet de prédire des structures secondaires sous-optimales. En revanche, la prédiction de complexes incluant des pseudonœuds n'est pas possible.

Dans la méthode de Tong et al. [216], la prédiction de structures secondaires d'un complexe d'ARN a été modélisée comme un problème d'interactions multiples d'ARN. Les auteurs ont montré que ce problème est NP-difficile avec prédiction ou non des pseudonœuds. Ils ont proposé plusieurs algorithmes d'approximations avec garanties de performances.

En 2016, la prédiction de structure secondaire de complexe d'ARN a été modélisée en tant qu'un problème d'optimisation combinatoire appelé *le problème des poteaux et des élastiques* [150]. Ce problème est connu et est adapté au problème biologique mais ne permet pas de modéliser les pseudonœuds. Les auteurs proposent également des algorithmes approchés avec garantie de performance pour résoudre ce problème.

En 2011 et 2014, deux outils, appelés NanoFolder et HyperFold [25], sont

développés. NanoFolder fonctionne en deux étapes : tout d'abord les hélices les plus probables sont calculées avec un modèle thermodynamique simple, puis un algorithme glouton permet de sélectionner et d'ajouter à la structure secondaire les hélices d'énergies minimales. HyperFold génère aussi toutes les hélices possibles mais utilise un algorithme plus sophistiqué que NanoFolder pour la sélection. Ces outils utilisent un modèle simple thermodynamique, ce qui leur permet de prédire des complexes incluant tout type de pseudonœuds. Cependant, ils ne peuvent pas retourner plusieurs solutions.

Parmi l'état de l'art, seuls les outils MultiRNAfold, NUPACK et NanoFolder sont disponibles. Les sources de NUPACK et MultiRNAfold sont disponibles, tandis que NanoFolder peut être utilisé via un webserver.

Un résumé des méthodes est donné dans le tableau 3.2.

Référence	Modèle	Outil	Structures sous-optimales	Pseudonœuds
Andronescu et al., 2005 [10]	MFE	MultiRNAfold		
Zadeh et al., 2011 [248]	MFE	NUPACK	✓	
Tong et al., 2014 [216]	MFE			✓
Mneimneh et Ahmed, 2016 [150]	MFE		✓	✓
Bindewald et al., 2016 [25]	MFE	NanoFolder		✓
Bindewald et al., 2016 [25]	MFE	HyperFold		✓

Tableau 3.2 : Résumé de l'état de l'art des outils de prédiction de structures secondaires d'un complexe d'ARN. Les outils en gris ne sont plus ou pas encore disponible, ou ils sont obsolètes.

3.2 Notre méthode : RCPred

Cette section est dédiée à la présentation de notre méthode pour prédire des structures secondaires de complexes d'ARN avec pseudonœuds internes et externes, retournant plusieurs structures. Dans un premier temps, nous présentons deux manières d'aborder le problème avec la programmation linéaire :

— La première méthode est une prédiction directe, c'est-à-dire que la prédiction du complexe d'ARN se fait en résolvant un seul programme linéaire, prenant en entrée uniquement des séquences d'ARN.

— La seconde méthode est une prédiction modulaire en deux étapes : la prédiction de plusieurs structures internes pour chaque ARN et de plusieurs interactions pour chaque paire d'ARN sont réalisées en amont, grâce à des outils de prédiction, puis la prédiction finale de la structure du complexe est réalisée grâce à un programme linéaire. Cette méthode est appelée RCPred, pour *RNA Complex Prediction*.

La seconde méthode permet de tirer parti des nombreux outils de prédiction de structures secondaires d'ARN et de prédiction d'interactions ARN-ARN. Elle permet également de rendre l'outil plus interactif en laissant la possibilité à l'utilisateur de faire varier les résultats en essayant différentes entrées.

La seconde méthode est basée sur le problème de détermination de la clique maximum. Ce problème étant NP-difficile, nous utilisons une heuristique, basée sur de la recherche locale, pour résoudre ce problème. Enfin, nous décrivons notre outil RCPred implémentant cette heuristique.

3.2.1 Un programme linéaire de prédiction d'appariements de complexes d'ARN

Nous présentons ici un programme linéaire pour prédire une structure secondaire d'un complexe d'ARN en maximisant la somme des probabilités d'appariement et des probabilités d'hybridation (associées aux appariements d'hybridation, c'est-à-dire des appariements entre deux ARN), tout en prenant en compte l'accessibilité des nucléotides. Ce programme linéaire est la généralisation du programme linéaire de l'outil RactIPAce [108] qui permet de prédire une structure jointe d'un duplexe d'ARN, avec le modèle MEA.

Soit r le nombre d'ARN. Soit m^a la longueur de l'ARN a, $1 \leq a \leq r$. Soient les variables de décisions binaires $x^a = (x^a_{ij})$ ($1 \leq a \leq r$, $1 \leq i \leq m^a - 1$, $i+1 \leq j \leq m^a$), représentant les appariements internes possibles de l'ARN a. Soient les variables de décisions binaires $z^{ab} = (z^{ab}_{ij})$ ($1 \leq a \leq r-1$, $a+1 \leq b \leq r$, $1 \leq i \leq m^a$, $1 \leq j \leq m^b$), représentant les appariements externes possibles des paires d'ARN a, b. On note p^a_{ij}, les probabilités d'appariements internes entre les bases i et j de l'ARN a ($1 \leq a \leq r$, $1 \leq i \leq m^a - 1$, $i+1 \leq j \leq m^a$). On note q^{ab}_{ij}, les probabilités d'appariements externes entre les bases i de l'ARN a et j de l'ARN b ($1 \leq a \leq r-1$, $a+1 \leq b \leq r$, $1 \leq i \leq m^a$, $1 \leq j \leq m^b$).

Pour empêcher les appariements non empilés, on définit les variables de décisions binaires za^{ab}_i et zb^{ab}_i ($1 \leq i \leq m^a$, $1 \leq a \leq r-1$, $a+1 \leq b \leq r$ et $1 \leq i \leq m^b$, $1 \leq a \leq r-1$, $a+1 \leq b \leq r$ respectivement). La variable za^{ab}_i est égale à 1 si la base i de l'ARN a est appariée avec une base de l'ARN b, 0 sinon. La variable zb^{ab}_i est égale à 1 si la base i de l'ARN b est appariée avec une base de l'ARN a, 0 sinon.

On note u^a_{ij}, les probabilités d'accessibilité des bases i à j pour l'ARN a ($i < j$). Soient les variables de décisions binaires v^a_{ij}, représentant l'accessibilité des bases i à j pour l'ARN a ($i < j$). Ces variables sont créées si les probabilités u^a_{ij} sont supérieures à un seuil. Si $v^a_{ij} = 1$, la séquence de i à j est accessible.

Soit v le nombre maximum de sites d'interaction pour chaque ARN.

Le programme linéaire est le suivant :

$$\max \sum_{a=1}^{r} \sum_{i=1}^{m^a-1} \sum_{j=i+1}^{m^a} (p^a_{ij} - \theta) x^a_{ij} + \alpha \sum_{a=1}^{r-1} \sum_{b=a+1}^{r} \sum_{i=1}^{m^a} \sum_{j=1}^{m^b} (q^{ab}_{ij} - \eta) z^{ab}_{ij} \tag{3.1}$$

tel que :

$$\sum_{j=1}^{i-1} x^a_{ji} + \sum_{k=i+1}^{m^a} x^a_{ik} + \sum_{l=1}^{m^b} z^{ab}_{il} \leq 1 \qquad 1 \leq i \leq m^a, 1 \leq a \leq r-1, a+1 \leq b \leq r \tag{3.2}$$

$$\sum_{j=1}^{i-1} x_{ji}^{a} + \sum_{k=i+1}^{m^a} x_{ik}^{a} + \sum_{l=1}^{m^b} z_{li}^{ab} \leq 1 \qquad 1 \leq i \leq m^a, 1 \leq a \leq r-1, a+1 \leq b \leq r \quad (3.3)$$

Ces contraintes 3.2 et 3.3 assurent qu'un nucléotide ne peut s'apparier qu'à un seul autre nucléotide. Les contraintes suivantes concernent les pseudo-nœuds.

$$x_{ij}^{a} + x_{kl}^{a} \leq 1 \qquad 1 \leq i < k < j < l \leq m^a, 1 \leq a \leq r \quad (3.4)$$

Si les deux séquences d'ARN sont données dans les sens 5' → 3' et 3' → 5', alors :

$$z_{ij}^{ab} + z_{kl}^{ab} \leq 1 \qquad 1 \leq i < k \leq m^a, \\ 1 \leq l < j \leq m^b, 1 \leq a \leq r-1, a+1 \leq b \leq r \quad (3.5)$$

Sinon si les deux séquences d'ARN sont données dans le même sens (5' → 3' et 5' → 3', ou bien, 3' → 5' et 3' → 5'), alors :

$$z_{ij}^{ab} + z_{kl}^{ab} \leq 1 \qquad 1 \leq i < k \leq m^a, \\ 1 \leq j < l \leq m^b, 1 \leq a \leq r-1, a+1 \leq b \leq r \quad (3.6)$$

Les contraintes 3.4 permettent d'interdire les pseudonœuds internes et les contraintes 3.5 et 3.6 les pseudonœuds externes (voir figure 3.1).

Les contraintes suivantes concernent l'accessibilité.

$$\sum_{j=1}^{i-1} x_{ji}^{a} + \sum_{j=i+1}^{m^a} x_{ij}^{a} + v_{kl}^{a} \leq 1 \qquad 1 \leq k < i < l \leq m^a, 1 \leq a \leq r \quad (3.7)$$

$$\sum_{j=1}^{m^b} z_{ij}^{ab} \leq \sum_{k \leq i \leq l} v_{kl}^{a} \qquad 1 \leq i \leq m^a, 1 \leq a \leq r-1, a+1 \leq b \leq r \quad (3.8)$$

$$\sum_{i=1}^{m^a} z_{ij}^{ab} \leq \sum_{k \leq j \leq l} v_{kl}^{b} \qquad 1 \leq j \leq m^b, 1 \leq a \leq r-1, a+1 \leq b \leq r \quad (3.9)$$

Les contraintes 3.7 permettent de définir si un nucléotide est apparié de façon interne : si oui, alors la région associée n'est pas accessible. Les contraintes 3.8 et 3.9 permettent de définir l'inverse : si un nucléotide est impliqué dans un appariement d'hybridation, la région associée est accessible.

$$v_{ij}^{a} + v_{kl}^{a} \leq 1 \qquad 1 \leq i < k \leq j < l \leq m^a, 1 \leq a \leq r \quad (3.10)$$

Les contraintes 3.10 interdisent le chevauchement de régions accessibles.

$$\sum_{i=1}^{m^a-1} \sum_{j=i+1}^{m^a} v_{ij}^{a} \leq v \qquad 1 \leq a \leq r \quad (3.11)$$

Les contraintes 3.11 permettent de limiter le nombre de sites d'interactions prédits. Cela pourrait être un paramètre choisi par l'utilisateur.

$$zb_i^{ab} = \sum_{j=1}^{m^a} z_{ji}^{ab} \qquad 1 \leq i \leq m^b, 1 \leq a \leq r-1, a+1 \leq b \leq r \tag{3.12}$$

$$za_i^{ab} = \sum_{j=1}^{m^b} z_{ij}^{ab} \qquad 1 \leq i \leq m^a, 1 \leq a \leq r-1, a+1 \leq b \leq r \tag{3.13}$$

$$zb_{i-1}^{ab} + (1 - zb_i^{ab}) + zb_{i+1}^{ab} \geq 1 \qquad 1 \leq i \leq m^b, 1 \leq a \leq r-1, a+1 \leq b \leq r \tag{3.14}$$

$$za_{i-1}^{ab} + (1 - za_i^{ab}) + za_{i+1}^{ab} \geq 1 \qquad 1 \leq i \leq m^a, 1 \leq a \leq r-1, a+1 \leq b \leq r. \tag{3.15}$$

Les contraintes 3.12 à 3.15 interdisent la prédiction d'appariements isolés.

Dans ce programme, θ et η sont des paramètres permettant de ne considérer que les probabilités élevées, et $\alpha \in (0,1)$ est un paramètre qui régule la proportion d'hybridation dans la structure.

Bien que ce programme linéaire a l'avantage de prédire la structure jointe d'un complexe à n ARN en résolvant un seul programme linéaire, elle présente aussi quelques désavantages. La complexité de résolution de ce programme linéaire en nombres entiers mono-objectif est exponentielle et dépend du nombre de variables. Si pour des duplexes de longueur raisonnable cette complexité reste abordable (voir comparaison des outils de prédiction de structures de duplexes [120]), elle devient vite ingérable lorsque l'on veut prédire des complexes à n ARN.

3.2.2 Approche modulaire et interactive de prédiction de structures de complexes d'ARN

Nous avons présenté précédemment un programme mono-objectif, mais nous nous intéressons plus particulièrement à des méthodes multi-objectifs afin de combiner plusieurs modèles. Or, comme nous l'avons observé avec l'outil BiokoP, cela augmente aussi la complexité de résolution des programmes, car des variables peuvent être ajoutées et surtout le programme doit être résolu plusieurs fois afin de trouver toutes les solutions du front de Pareto (et des k fronts de Pareto pour déterminer des solutions sous-optimales).

Enfin, nous voulons proposer une méthode interactive et modulaire afin de laisser à l'utilisateur le choix d'orienter les prédictions et de choisir différents modules qui seront les outils de prédictions des structures secondaires d'ARN seuls et des sites d'interactions. Par ailleurs, une méthode modulaire permet de répartir les calculs sur différentes machines si ceux-ci requièrent des moyens importants.

Ces objectifs ne sont pas réalisables avec la méthode présentée précédemment et c'est pourquoi nous proposons ici une méthode plus adaptée.

De nombreux outils existent pour prédire la structure secondaire des ARN (voir section 2.1) ainsi que pour prédire la structure d'interactions ARN-ARN (voir section 3.1.1). C'est à partir de ce constat que nous avons imaginé la

solution suivante pour prédire des complexes de plus de deux ARN. La prédiction d'un complexe d'ARN peut être vue comme la détermination de la combinaison, ayant l'énergie minimum, de structures et d'interactions parmi un ensemble de structures secondaires prédites pour chaque ARN et parmi un ensemble d'interactions prédites pour chaque paire d'ARN.

Notre méthode prend en entrée, pour un ensemble d'ARN d'intérêt, un ensemble de structures secondaires pour chaque ARN et un ensemble de sites d'interaction par paire d'ARN. Elle prédit un ensemble de complexes possibles de basses énergies, constitués des structures et interactions d'entrée. Les structures secondaires en entrée peuvent contenir des pseudonœuds internes (voir figure 3.1 A), de même que les interactions peuvent contenir des pseudonœuds externes (voir figure 3.1 B). Des structures secondaires avec pseudonœuds peuvent être obtenues par des outils comme BiokoP [122], pKiss [101], McGenus [30]. En revanche, nous ne connaissons aucun outil de prédiction d'interactions prédisant des pseudonœuds externes.

Le principe de notre méthode modulaire, appelée RCPred (*RNA Complex Prediction*), est illustrée dans la figure 3.2.

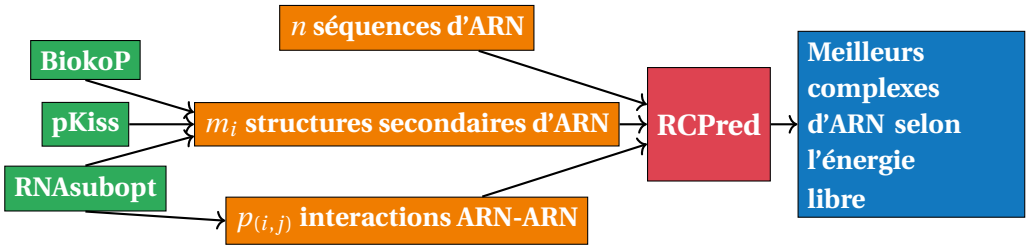

Figure 3.2 : Principe général de notre méthode RCPred. Illustration avec comme outils de prédiction de structures secondaires : BiokoP [122], pKiss [101], RNAsubopt [130], et comme outil de prédiction d'interaction ARN-ARN : RNAsubopt [130].

3.2.3 Modélisation par une clique pondérée

Dans cette section, nous montrons que la prédiction de complexes peut être modélisée comme un problème d'optimisation combinatoire sur les graphes. Les structures secondaires et les interactions en entrée sont les sommets de ce graphe. Chaque sommet est pondéré par l'énergie libre correspondante. Si deux structures secondaires (ou deux interactions ou une structure et une interaction) peuvent former un complexe sans qu'il y ait de conflits entre les appariements, elles sont dites *compatibles*. Cela se traduit sur le graphe par une arête entre les deux sommets. Le problème consiste alors à trouver le sous-graphe de poids minimum où tous les sommets sont liés les uns aux autres, c'est-à-dire à trouver une clique de poids minimum (voir définition 1.2.7, page 24). Classiquement, le problème de la clique pondérée est un problème de maximisation. Dans la suite, le problème sera décrit de manière à maximiser la somme des poids où le signe des poids sera inversé.

Problème de la clique pondérée contrainte

Soit un graphe pondéré $G(V,E)$ que nous appellerons par la suite graphe de compatibilité, tel que :

— V, l'ensemble des sommets, est composé de deux sous-ensembles, V^S et V^I, où V^S est l'ensemble des sommets représentant les structures secondaires, et V^I est l'ensemble des sommets représentant les interactions. Chaque sommet $v \in V$ a un poids égal à l'opposé de l'énergie associée à la structure ou à l'interaction.

— E, l'ensemble des arêtes, représente les compatibilités entre les sommets. Une arête existe, si et seulement si, deux sommets sont compatibles. Nous considérons que deux sommets ne sont pas compatibles s'il y a au moins deux nucléotides identiques impliqués dans différents appariements. Ces règles de compatibilité permettent la présence de n'importe quel motif dans les complexes : les pseudonœuds internes (qui peuvent également être déjà présents dans les structures secondaires d'entrée) et les pseudonœuds externes.

Un complexe d'ARN peut être vu comme une clique contrainte car pour chaque ARN, il doit y avoir au plus une structure secondaire par ARN.

Le poids d'une clique (contrainte ou non) est la somme des poids de ses sommets. Le problème de la clique contrainte consiste à trouver une clique telle qu'elle soit composée d'au plus une structure secondaire par ARN et que son poids soit minimum.

Calcul de l'énergie libre

Le poids de chaque sommet du graphe représente l'opposé de l'énergie libre de la structure ou de l'interaction associée. Cette énergie est calculée de manière à unifier les différentes sources de prédiction des structures secondaires et des interactions. Nous avons utilisé deux modèles pour calculer cette énergie :

— Le premier modèle est le modèle de Turner [218] (avec les paramètres énergétiques de la mise à jour de 2004), qui est utilisé pour les structures secondaires sans pseudonœuds et pour les interactions.

— Le second modèle est basé sur la somme des énergies des empilements d'appariements (à partir du modèle de Turner), utilisé dans l'outil BiokoP. Ce modèle est utilisé pour le calcul de l'énergie des structures secondaires avec pseudonœuds.

L'opposé de l'énergie d'un complexe est ensuite approximée en faisant la somme des poids des sommets appartenant à la clique correspondante à ce complexe.

Un désavantage de cette méthode est qu'elle ne permet pas de prend en compte le coût entropique lors de la formation de nouveaux pseudonœuds [33] ou de nouvelles interactions [63], cela est discuté à la fin de ce chapitre.

3.2.4 Résolution exacte du problème de la clique pondérée

Le problème de la clique est NP-difficile [107]. Ce problème est très étudié et de nombreuses méthodes existent pour le résoudre. Les méthodes exactes, qui permettent de trouver la solution optimale par l'optimisation du poids de la clique, sont soit des généralisations de méthodes pour le problème non-pondéré [118, 158], soit des algorithmes de type *branch and bound* [229].

Le problème de la clique est un problème difficile à approximer. Il n'existe pas d'algorithme polynomial permettant de trouver de meilleures solutions que $(n^{1-\epsilon})$-approchées, pour tout $\epsilon > 0$ [89]. A fortiori, il n'existe pas d'algorithmes polynomiaux permettant de résoudre des problèmes de cliques multi-critères.

La résolution exacte de ce problème peut être réalisée en le modélisant comme un programme linéaire en nombres entiers puis en faisant appel à un solveur. Le programme linéaire est donné ci-dessous en fonction du graphe de compatibilité $G(V,E)$ et de l'ensemble d'ARN d'intérêts, noté N. Les variables de décisions x_v $(v \in V)$ sont telles que $x_v = 1$ si le sommet v fait partie de la clique, et $x_v = 0$ sinon.

$$\max \sum_{v \in V} w_v x_v \tag{3.16}$$

tel que :

$$\sum_{u \in \overline{E_v}} x_u + |\overline{E_v}| x_v \leq |\overline{E_v}| \qquad \forall v \in V \tag{3.17}$$

$$\sum_{v \in V_a^S} x_v \leq 1 \qquad \forall a \in N \tag{3.18}$$

$$x_v \in \{0,1\} \qquad \forall v \in V. \tag{3.19}$$

où w_v désigne le poids du sommet v, $\overline{E_v}$ est l'ensemble des sommets non-adjacents au sommet v, i.e. $\overline{E_v} = \{u | u \in V, (u,v) \notin E\}$, $|\overline{E_v}|$ est le cardinal de $\overline{E_v}$, et $V_a^S \subset V$ est l'ensemble des sommets structures secondaires concernant l'ARN a, i.e. $V_a^S = \{v | v \in V^S$, v est une structure secondaire de l'ARN $a\}$.

La fonction objectif 3.16 maximise la somme des poids (de signe opposé) des sommets de la clique. Les contraintes 3.17 forcent l'ensemble des sommets sélectionnés pour faire partie de la solution à être une clique. Les contraintes 3.18 empêchent qu'il y ait plus d'une structure secondaire pour chaque ARN $a \in N$ qui sera dans la clique.

Afin de respecter la contrainte d'avoir au plus une structure secondaire par ARN, nous pouvons également apporter une modification sur la définition du graphe G, où les arêtes entre les sommets de structures sont définies de la manière suivante : si deux structures secondaires concernent le même ARN, alors elles ne sont pas compatibles ; dans le cas contraire, elles sont compatibles. Ainsi, on se ramène à un problème de clique non contraint, où le programme linéaire, auquel nous nous référons par PL1, est tel que :

$$\max \sum_{v \in V} w_v x_v \tag{3.20}$$

tel que :

$$\sum_{u \in \overline{E_v}} x_u + |\overline{E_v}| x_v \leq |\overline{E_v}| \qquad \forall v \in V \tag{3.21}$$

$$x_v \in \{0,1\} \qquad \forall v \in V. \tag{3.22}$$

3.2.5 Résolution approchée du problème de la clique pondérée : utilisation d'une heuristique

La complexité de résolution d'un programme entier est exponentielle en temps. Par ailleurs, afin de générer des solutions sous-optimales, le programme doit être résolu plusieurs fois avec une complexité qui augmente à chaque nouvelle résolution.

De par la nature NP-difficile du problème, de nombreuses heuristiques ont été proposées. Les heuristiques proposées dans la littérature sont basées sur la recherche locale [172], la recherche tabou [239], ou sur d'autres techniques [17, 202, 137].

Nous proposons ici une adaptation de l'heuristique *Breakout Local Search* (BLS) [17] pour déterminer des solutions supplémentaires à la solution maximale. Nous présentons tout d'abord l'heuristique BLS puis les modifications que nous avons apportées à l'heuristique pour notre problème de prédiction de complexes d'ARN.

Breakout Local Search

L'heuristique BLS [17] a été proposée par Benlic and Hao pour le problème de la clique pondérée et est basée sur la recherche locale et tabou.

La recherche locale [2] est une heuristique très générale qui permet de trouver de bonnes solutions pour de nombreux problèmes d'optimisation combinatoire. Il s'agit d'une méthode itérative. La recherche locale démarre avec une solution initiale et elle la modifie à chaque étape en explorant le voisinage de la solution. Le voisinage est un ensemble de solutions voisines obtenues par de petites modifications, appelées mouvements, sur la solution courante. Quand une solution ne peut plus être améliorée, il s'agit d'un optimum local.

La recherche tabou [239] est une métaheuristique basée sur la recherche locale. La différence principale avec la recherche locale est qu'à chaque itération, la meilleure solution voisine est sélectionnée, même si elle n'est pas meilleure que la solution courante. Dans le but d'éviter de trouver des solutions déjà obtenues auparavant, une liste tabou est utilisée.

L'heuristique BLS commence avec une solution aléatoire et réalise ensuite alternativement deux phases jusqu'à atteindre la limite du temps d'exécution. Les deux phases sont les suivantes :

1. Recherche locale : on effectue une recherche locale jusqu'à trouver un optimum local.
2. Perturbation : on modifie significativement l'optimum local afin de s'en échapper et d'explorer plus amplement l'espace de recherche.

Durant la phase de recherche locale, tous les mouvements possibles sont considérés et celui optimisant le plus le poids de la solution est choisi. Les mouvements possibles sont : ajouter ou remplacer un sommet de la clique. Afin de définir les mouvements, les définitions suivantes sont requises. Soit $G(V,E)$ un graphe et C la clique courante (voir figure 3.3) :

— PA est l'ensemble des sommets qui peuvent être directement ajoutés à la clique C : autrement dit, il existe des arêtes entre tous ces sommets et tous les sommets de la clique C ; $PA = \{v : v \notin C, \forall u \in C\ [v,u] \in E\}$.

— OM est l'ensemble des paires de sommets (v,u) où v n'est pas dans la clique C mais où il existe des arêtes entre v et tous les sommets de la clique C sauf avec le sommet u ; $OM = \{(v,u) : v \notin C$ et $u \in C, \forall v' \in C \backslash \{u\}\ [v,v'] \in E$ et $[v,u] \notin E\}$. L'ensemble OM est utilisé pour effectuer les mouvements de remplacement.

— OC est l'ensemble des sommets n'étant pas dans la clique C : $OC = \{v \in V \backslash C\}$.

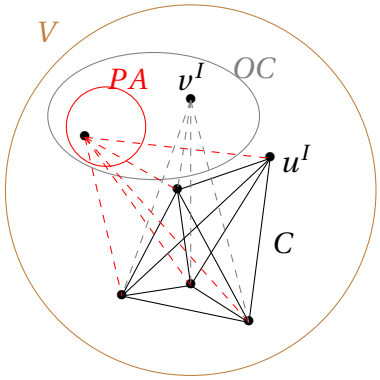

Figure 3.3 : Ensembles utilisés dans l'heuristique *breakout local search*. C est une clique, PA est l'ensemble composé des sommets tous liés à tous les sommets de la clique, OC est l'ensemble des sommets n'étant pas dans la clique.

La phase de perturbation vise à modifier la solution courante afin de s'échapper d'un minimum local. Cette phase peut dégrader fortement la solution. La force de dégradation dépend du nombre de fois où la solution n'a pas été améliorée lors de la phase de recherche locale. Les stratégies de perturbation sont basées sur quatre mouvements qui sont réalisés plusieurs fois : ajout, remplacement ou suppression d'un sommet de la clique (perturbation faible) et réinitialisation de la clique (perturbation forte). Durant cette phase, une liste tabou est utilisée pour éviter l'utilisation d'un sommet lors d'un mouvement, si celui-ci a été supprimé de la solution précédemment. Cette phase de perturbation est une différence importante avec les autres méthodes de recherche locale : elle permet d'explorer plus efficacement et plus rapidement l'espace de recherche en évitant de rester bloqué dans un optimum local. La meilleure solution trouvée pendant l'exécution de l'heuristique est enregistrée et retournée à la fin.

Benlic et Hao ont montré que l'heuristique BLS permet d'obtenir de meilleurs résultats que ceux connus jusqu'alors pour plusieurs instances du problème

pondéré de la clique. Ils ont montré également que cette heuristique est utilisable pour des graphes de grande taille (jusqu'à 3300 sommets et 5000000 arêtes) en un temps raisonnable.

Adaptation de l'heuristique BLS

Nous présentons ici l'adaptation de la méthode BLS à notre problème de clique pondérée. Nous devons adapter l'heuristique afin de pouvoir générer un ensemble de cliques et non plus une seule. De plus, nous modifions la condition d'arrêt afin que l'algorithme se termine de manière non aléatoire. Nous modifions également la génération de clique initiale afin de la rendre plus simple.

Les modifications apportées à l'algorithme sont décrites ci-dessous :

— **Génération de la clique initiale :** dans l'heuristique BLS, cette procédure consiste à sélectionner aléatoirement un sommet et ensuite à lui ajouter itérativement des sommets qui peuvent former une clique, jusqu'à ce qu'il n'y ait plus d'autres sommets qui puissent être ajoutés. Dans notre algorithme, cette procédure consiste à sélectionner aléatoirement un unique sommet constituant une clique. La clique grossira ensuite grâce à la phase de recherche locale, ce qui permet d'avoir une solution meilleure au départ que si les sommets étaient choisis aléatoirement.

— **Génération de plusieurs cliques :** dans notre algorithme, chaque optimum local trouvé est sauvegardé. Quand les solutions sont retournées à la fin de l'algorithme, elles sont triées selon leur énergie.

— **Condition d'arrêt de la recherche :** dans l'heuristique BLS, la condition d'arrêt est un nombre d'itérations maximum. Dans notre algorithme nous avons choisi d'arrêter la recherche lorsque nous constatons que celle-ci stagne, plus précisément, lorsque la recherche est bloquée dans un optimum local (c'est-à-dire qu'on ne peut plus trouver de solution meilleure que la solution courante) et que malgré deux perturbations aléatoires, on ne réussi pas à s'échapper de l'optimum local.

L'heuristique BLS nécessite un ensemble de paramètres permettant de moduler la force de la phase de perturbation (L_0 et L_{Max}), le nombre maximum d'itérations pendant lesquelles la solution n'est pas améliorée (T), des coefficients pour accepter des solutions non-améliorées (α_s et α_r), un coefficient pour la liste tabou (ϕ) et la probabilité d'appliquer la perturbation dirigée (P_0). Nous utilisons les paramètres fixés par les auteurs de l'heuristique BLS sur un jeu de données de référence pour les graphes donnés par le site web DIMACS (`http://dimacs.rutgers.edu/index.php/`) : $L_0 = 4$, $L_{Max} = 4$, $T = 1000$, $\alpha_s = 0.7$, $\alpha_r = 0.92$, $\phi = 7$ et $P_0 = 0.75$.

3.2.6 Outil RCPred

Nous avons implémenté notre méthode en C++. RCPred prend en entrée n séquences d'ARN, plusieurs structures secondaires par ARN et plusieurs interactions par paire d'ARN. Les compatibilités entre les structures secondaires et les interactions sont déterminées et le graphe est construit. L'heuristique retourne ensuite des cliques desquelles sont dérivés des complexes d'ARN.

Si certaines séquences sont identiques, des complexes symétriques peuvent être générés. Ils sont alors identifiés et supprimés pour éviter la redondance dans les résultats. Enfin les complexes d'ARN sont triés selon leur énergie et retournés à l'utilisateur.

RCPred est disponible sur la plateforme EvryRNA qui permet une certaine liberté à l'utilisateur. Ce dernier peut choisir quels outils seront utilisés pour prédire les structures secondaires et les sites d'interactions qui seront donnés en entrée à RCPred. Les paramètres pour moduler le nombre de solutions retournées par ces outils sont également modifiables (figure 3.4 A). Une fois que l'utilisateur a rempli ce premier formulaire, les résultats des outils choisis sont affichés pour chaque ARN et chaque paire d'ARN. L'utilisateur peut modifier ces résultats s'il possède déjà des connaissances sur quelques structures ou interactions (figure 3.4 B). Il peut également supprimer complètement certains résultats et en ajouter d'autres. Cela permet d'aiguiller l'outil vers de meilleures prédictions. Sur la dernière page sont affichés les résultats, les complexes sont représentés grâce à l'outil forna [109] qui permet une certaine interactivité avec l'utilisateur (figure 3.4 C). D'autres outils de visualisation interactifs existent, comme par exemple VARNA [48] ou Ribosketch [133]. Ribosketch a été développé après RCPred est permet une meilleure visualisation des complexes, il a été intégré dans l'outil C-RCPred présenté dans le chapitre 4.

3.3 Résultats

Cette section est dédiée à la présentation des résultats de l'outil RCPred (*RNA Complex Prediction*) qui implémente l'heuristique décrite précédemment pour prédire des structures secondaires de complexes d'ARN. Dans un premier temps, nous évaluons l'heuristique séparément sur un jeu de données de graphes. Nous étudions ensuite la qualité des entrées de RCPred : les structures secondaires et les interactions. Enfin, nous évaluons l'outil RCPred et le comparons aux outils de la littérature sur un jeu de données de complexes d'ARN dont la structure secondaire est connue.

3.3.1 Jeu de données

Pour évaluer RCPred, nous avons établi un jeu de données, noté C_{multi} composé de 90 complexes d'ARN, provenant de la base de données RNA STRAND [11]. Cette base de données rassemble des structures provenant d'autres bases de données comme par exemple dans notre cas la Protein Data Bank [19] ou la Nucleic Acid Database [18]. Ces bases rassemblent des structures 3D qui sont validés expérimentalement soit par RMN, soit par cristallographie à rayons X, pour les structures que nous avons récupérées. Les structures secondaires ont été déterminées grâce à l'outil RNAview [247]. Tous les complexes de C_{multi} ne contiennent aucun nucléotide modifié et ont une longueur variant entre 20 et 1000 nucléotides.

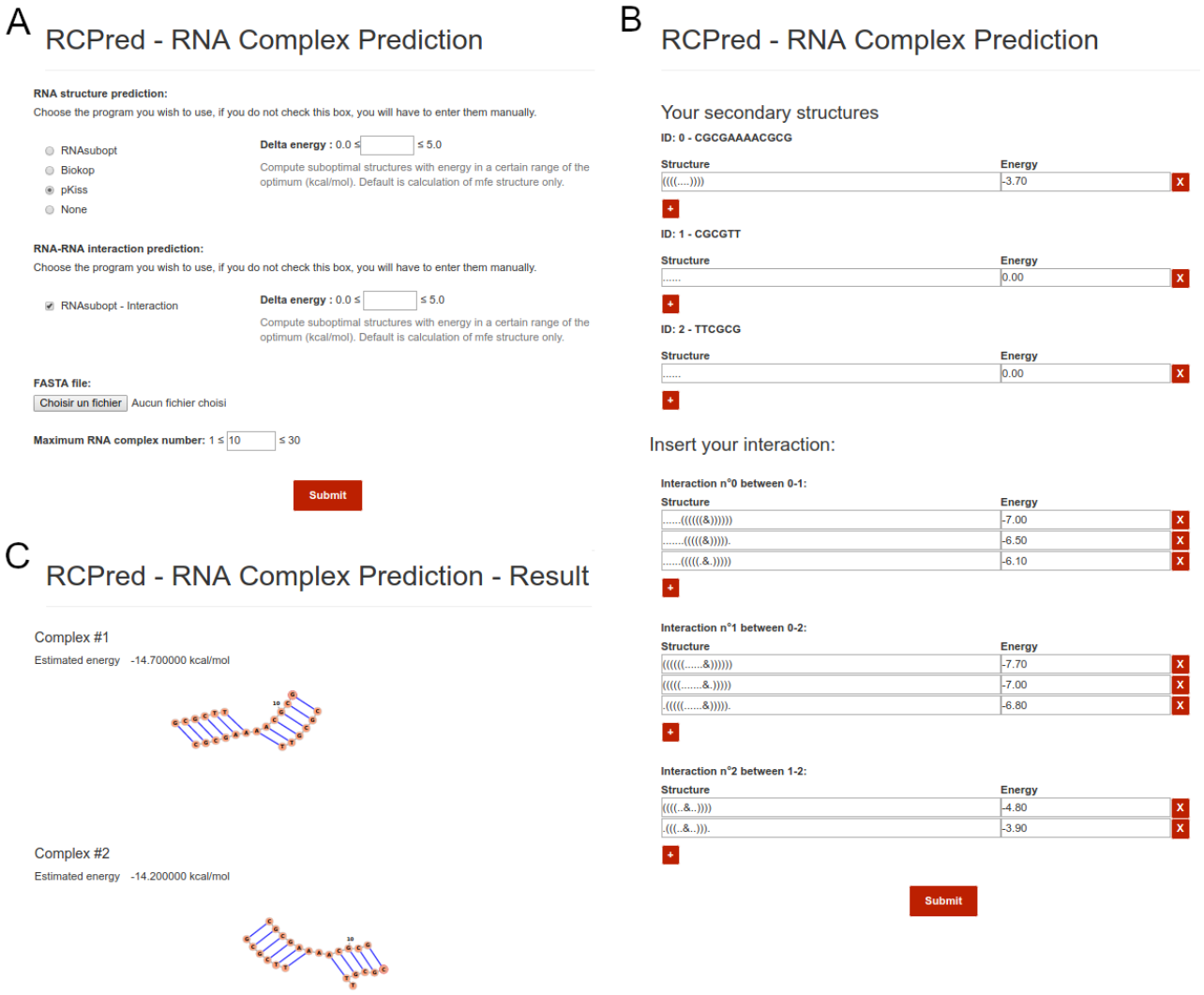

Figure 3.4 : RCPred sur la plateforme EvryRNA. Premier formulaire (A), second formulaire (B) et résultats (C).

3.3.2 Évaluation de la performance de l'heuristique

Nous étudions ici les performances de l'heuristique par rapport à la résolution du programme linéaire PL1 (décrit en section 3.2.4) sur le jeu de données C_{multi} : sa capacité à trouver les solutions exactes, les temps d'exécutions.

Pour évaluer la capacité de l'heuristique à trouver les solutions exactes, nous résolvons PL1 pour obtenir la solution exacte et notons le temps mis. Puis nous exécutons l'heuristique 10 fois pour obtenir le nombre moyen de fois où la solution exacte est trouvée ainsi que le temps moyen mis. Ces résultats sont présentés dans les figures 3.5 et 3.6, où les graphes sont triés suivant leur nombre d'arêtes.

Comme nous pouvons le voir, les poids des cliques trouvées par l'heuristique sont optimaux dans la plupart des cas (figure 3.5). Pour 3 graphes seulement, l'heuristique obtient des cliques ayant un écart supérieur à 5 kcal.mol^{-1}, la différence maximale avec la clique maximum est de 18 kcal.mol^{-1}. Ces trois graphes correspondent à des complexes d'ARN longs et donc en proportion,

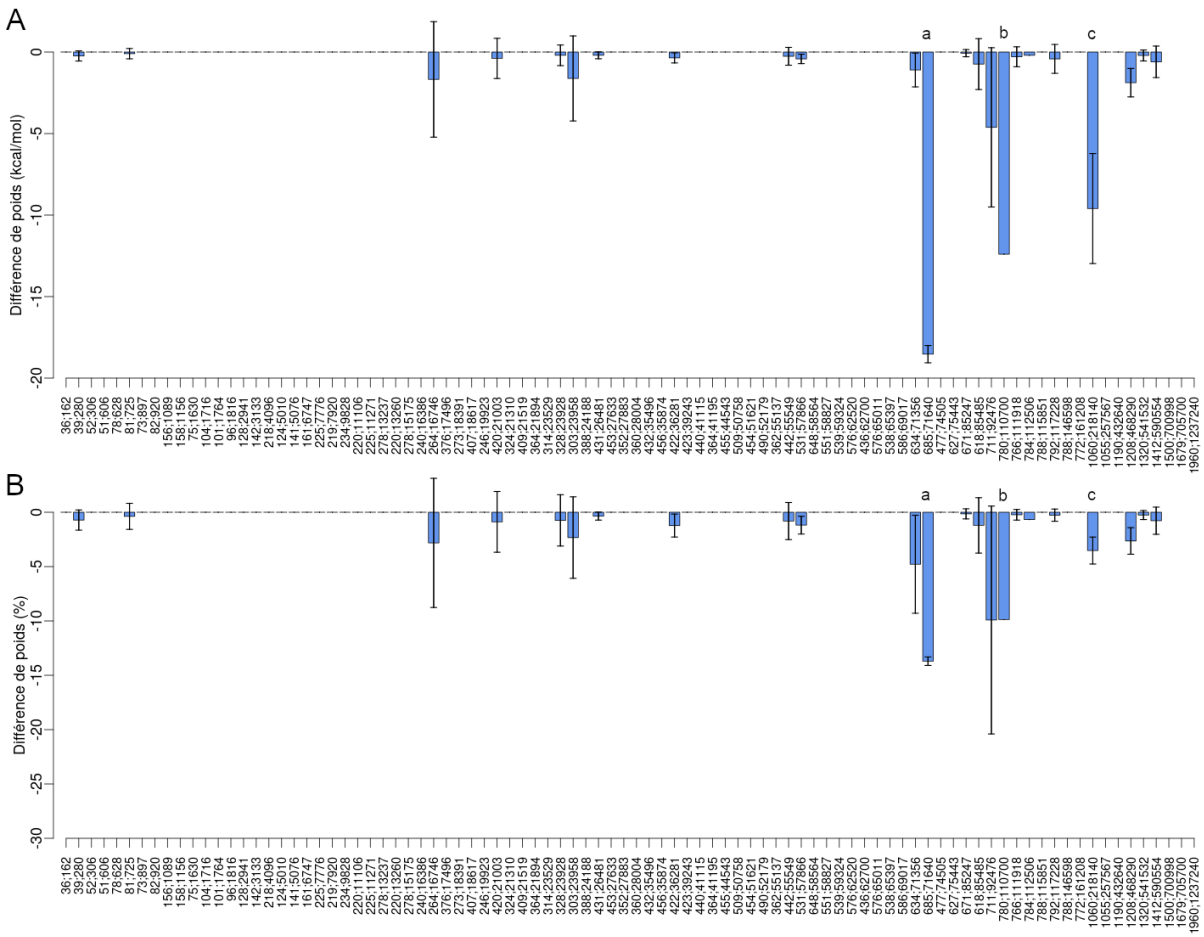

Figure 3.5 : Performance de l'heuristique : poids de la clique optimale. Différence de poids moyenne en kcal.mol^{-1} (A) ou en pourcentage (B) entre la clique optimale et la clique trouvée par l'heuristique. Les graphes testés sont ceux élaborés à partir des entrées générées pour le jeu de données C_{multi} (voir section 3.3.3). En abscisse est notée pour chaque graphe le nombre de sommets et le nombre d'arêtes. Les graphes sont triés suivant le nombre d'arêtes. L'heuristique est exécutée 10 fois pour obtenir les moyennes des poids des cliques et des temps. La barre d'erreur indique l'écart-type sur le poids moyen de l'heuristique.

les différences reviennent à 13% (graphe a, 272 nucléotides), 10% (graphe b, 152 nucléotides), et 3% (graphe c, 545 nucléotides). Nous pouvons voir que pour seulement 4 graphes l'écart-type se situe entre 2 et 5 kcal.mol^{-1} (figure 3.5 A). Tous les autres écart-types sont inférieurs à 2. Cela montre que la variabilité entre les exécutions de l'heuristique est très faible.

Le temps d'exécution de l'heuristique est de quelques secondes tout au plus. La figure 3.6 reporte la différence de temps d'exécution entre la résolution de PL1 et de l'heuristique pour chaque graphe. Les graphes sont triés suivant leur nombre d'arêtes car c'est ce qui est déterminant pour la complexité de résolution de PL1. Le nombre de sommets rentre également en jeu pour la complexité de résolution de PL1, mais les graphes testés ont un nombre de

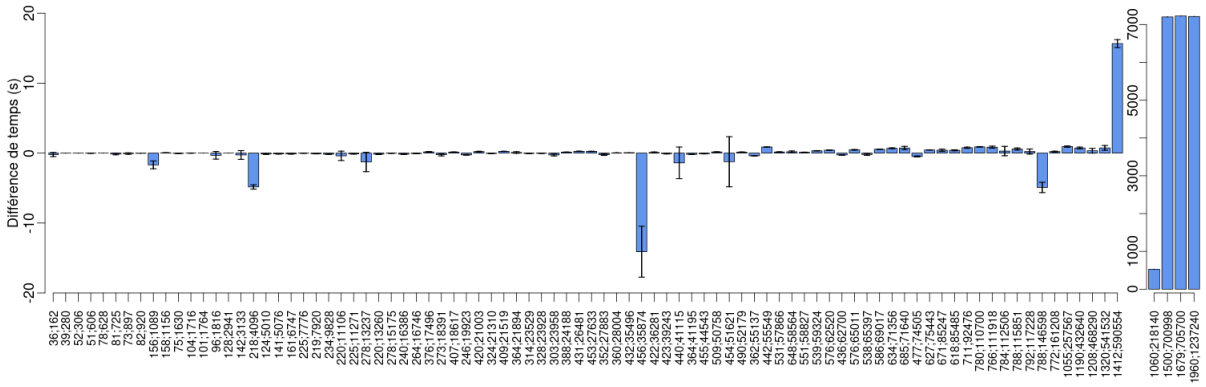

Figure 3.6 : Performance de l'heuristique : temps d'éxecution. Les graphes testés sont ceux élaborés à partir des entrées générées pour le jeu de données C_{multi} (voir section 3.3.3). Différence de temps d'exécution entre la résolution de PL1 et l'heuristique. En abscisse est notée pour chaque graphe le nombre de sommets et le nombre d'arêtes. Les graphes sont triés suivant le nombre d'arêtes. L'heuristique est exécutée 10 fois pour obtenir les moyennes des poids des cliques et des temps. La barre d'erreur indique l'écart-type sur le temps moyen de l'heuristique.

sommets plus homogène que celui des arêtes, ce qui ne permet pas de tester ce paramètre. Pour résoudre PL1, nous avons utilisé comme solveur CPLEX version 12.63 [96] et avons limité le temps de résolution à deux heures.

Comme attendu, les temps d'exécution de l'heuristique sont bien meilleurs lorsque les graphes sont importants. Les temps d'exécution de la résolution de PL1, pour les plus petits graphes, sont similaires à ceux de l'heuristique. Suivant la taille du graphe, nous pouvons alors proposer à l'utilisateur une résolution du problème de façon exacte ou avec l'heuristique. De plus, les écart-types sont très faibles, donc la variabilité du temps d'exécution de l'heuristique est très faible.

Il faut aussi rappeler que l'heuristique permet de retourner plusieurs solutions en une seule exécution, ce qui n'est pas le cas en résolvant le programme linéaire. En effet, il faudrait résoudre PL1 autant de fois que l'on veut de solutions et donc le temps serait multiplié par ce nombre.

3.3.3 Qualité des entrées de RCPred

Nous étudions ici la qualité des structures secondaires et des interactions utilisées en entrée de RCPred pour le jeu de données C_{multi}.

Pour chaque ARN du jeu de données C_{multi}, nous avons généré des structures secondaires en utilisant trois outils, à savoir BiokoP, pKiss [101] et RNAsubopt du ViennaRNA package [130]. Ces outils permettent de générer des structures sous-optimales et BiokoP et pKiss permettent de prédire les pseudonœuds. De plus ces outils font partie des outils ayant les meilleures performances. Pour chaque outil, nous avons prédit au maximum 30 structures. Nous avons utilisé ces résultats en entrée dans RCPred, soit en les utilisant tous, soit indépendamment. Nous avons donc testé RCPred, avec comme structures

secondaires en entrée :

— soit les prédictions de BiokoP, pKiss et RNAsubopt ensemble,
— soit les prédictions de BiokoP,
— soit les prédictions de pKiss,
— soit les prédictions de RNAsubopt.

Les interactions entre les paires d'ARN ont été générées grâce à RNAsubopt. Le nombre de structures sous-optimales a été fixé à 90.

Comme nous pouvons le voir sur la figure 3.7 A, la qualité des structures secondaires est assez disparate. En effet, l'écart entre les MCC minimaux et maximaux est important. La plupart des structures secondaires utilisées sont assez éloignées de la structure de référence pour chaque complexe, avec un MCC médian aux environs de 20. Cela se vérifie pour tous les outils utilisés : BiokoP, pKiss et RNAsubopt (figure 3.8 A, B et C). La proportion de "bonnes" structures est faible pour tous les complexes. Cela s'explique en partie par le fait que les ARN formant des complexes sont engagés pour la plupart dans des interactions, ce qui réduit le nombre d'appariements intra-ARN. D'autre part, les outils utilisés ne tenant pas compte d'information comme l'accessibilité pour la prédiction, cela est normal d'obtenir des prédictions éloignées de la structure de référence.

Nous pouvons néanmoins observer sur la figure 3.8, que RNAsubopt semble donner de meilleures prédictions que les autres outils, viennent ensuite BiokoP puis pKiss (d'après les MCC moyens et médians). En effet, la plupart des structures de ses ARN ne possède pas de pseudonœuds. Les résultats donnés par les différents outils sont complémentaires. L'important pour notre outil est d'obtenir un ensemble diversifié de structures contenant les structures réelles (ce qui est le cas quand nous observons les MCC maximaux) afin de prédire la structure des complexes.

La qualité des interactions utilisées en entrée est visible en figure 3.7 B. Ces interactions sont générées par l'outil RNAsubopt et comme nous pouvons le voir, la qualité de ces interactions est meilleure que celle des structures secondaires, avec un MCC médian aux environ de 80, quelle que soit la longueur des complexes. La qualité des interactions est, comme pour les structures secondaires, assez inégale entre les interactions : l'écart entre les MCC minimaux et maximaux est important pour tous les complexes.

3.3.4 Évaluation de RCPred

Cette section est dédiée à l'analyse des résultats obtenus par RCPred. Nous étudions tout d'abord la qualité des structures sous-optimales prédites et l'utilité de leur prédiction. Puis nous étudions l'influence que la qualité des entrées de RCPred peut avoir sur les prédictions. Les résultats de RCPred sont des moyennes obtenues sur 10 exécutions.

Importance de générer plusieurs structures

Nous présentons ici les résultats des 10 premières structures retournées par RCPred pour chaque complexe de C_{multi}. Les MCC moyens (sur 10 itérations) sont représentés en figure 3.9. Les structures utilisées en entrée sont celles

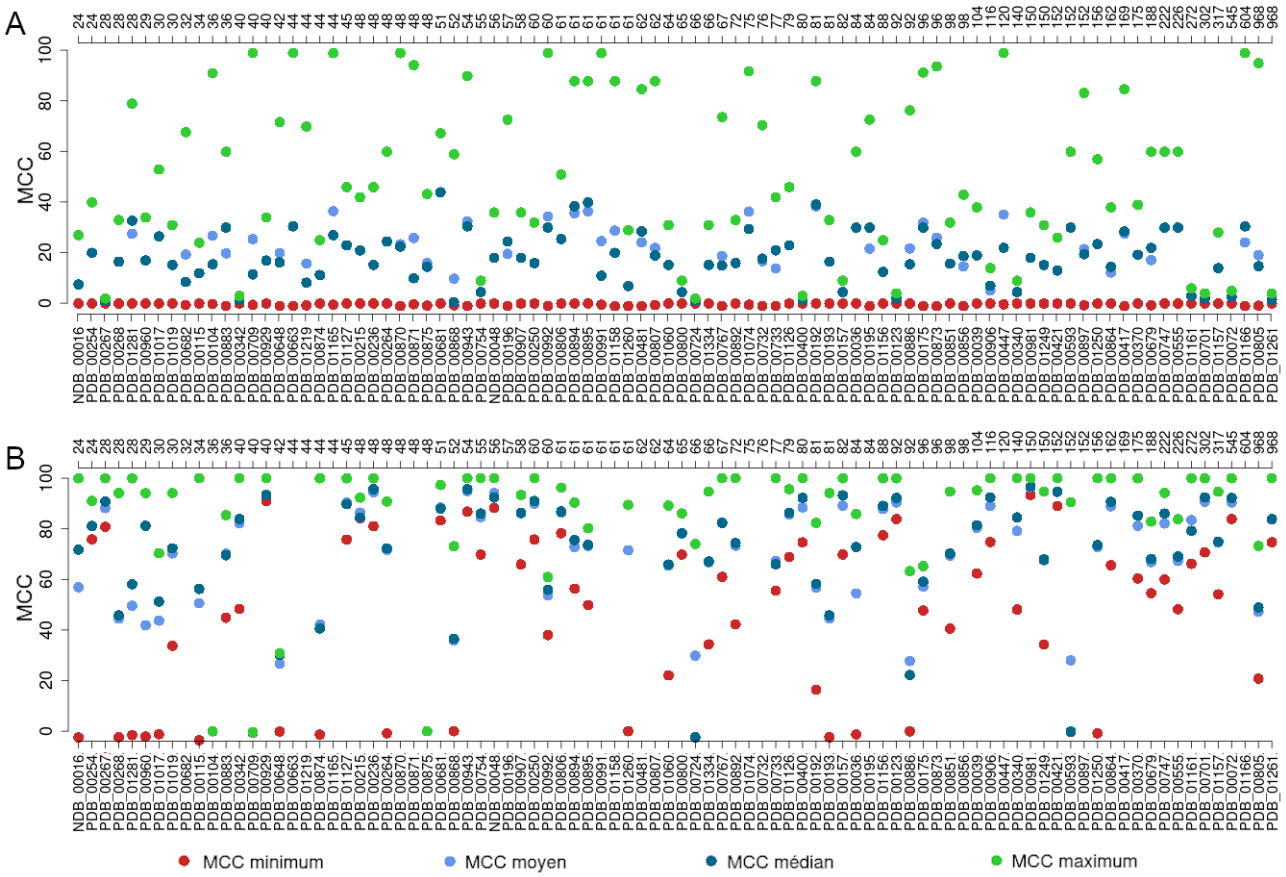

Figure 3.7 : MCC des structures et interactions d'entrée de RCPred pour chaque complexe de C_{multi}. La longueur des complexes est indiquée par l'axe des ordonnées supérieur. A) Moyenne, médiane, minimum et maximum MCC sur l'ensemble des structures secondaires d'entrée de RCPred, générées conjointement par BiokoP, pKiss et RNAsubopt. B) Moyenne, médiane, minimum et maximum MCC sur l'ensemble des interactions d'entrée de RCPred, générées par RNAsubopt.

générées par BiokoP, pKiss et RNAsubopt. Comme nous pouvons le voir, la plupart du temps, la meilleure structure (ayant le meilleur MCC) est la première solution retournée, c'est-à-dire celle ayant l'énergie libre la plus basse. Cela concerne 65% des complexes.

Pour 35% des complexes, la meilleure structure n'est pas la première solution retournée par l'heuristique : par exemple pour le complexe PDB_00733, de 77 nucléotides, c'est la sixième solution retournée qui est la plus proche de la structure de référence (MCC égal à environ 70%). Parmi ces complexes, la meilleure solution est le plus souvent la deuxième solution retournée : pour environ 13% des complexes. Les meilleures solutions restantes sont réparties aléatoirement entre la troisième et la septième solutions retournées.

En revanche, si nous observons la totalité des complexes, nous pouvons voir que la qualité des solutions augmente à mesure que l'énergie libre diminue. Ce graphe montre, d'une part, l'utilité de prédire plusieurs structures et, d'autre part, valide le modèle énergétique utilisé.

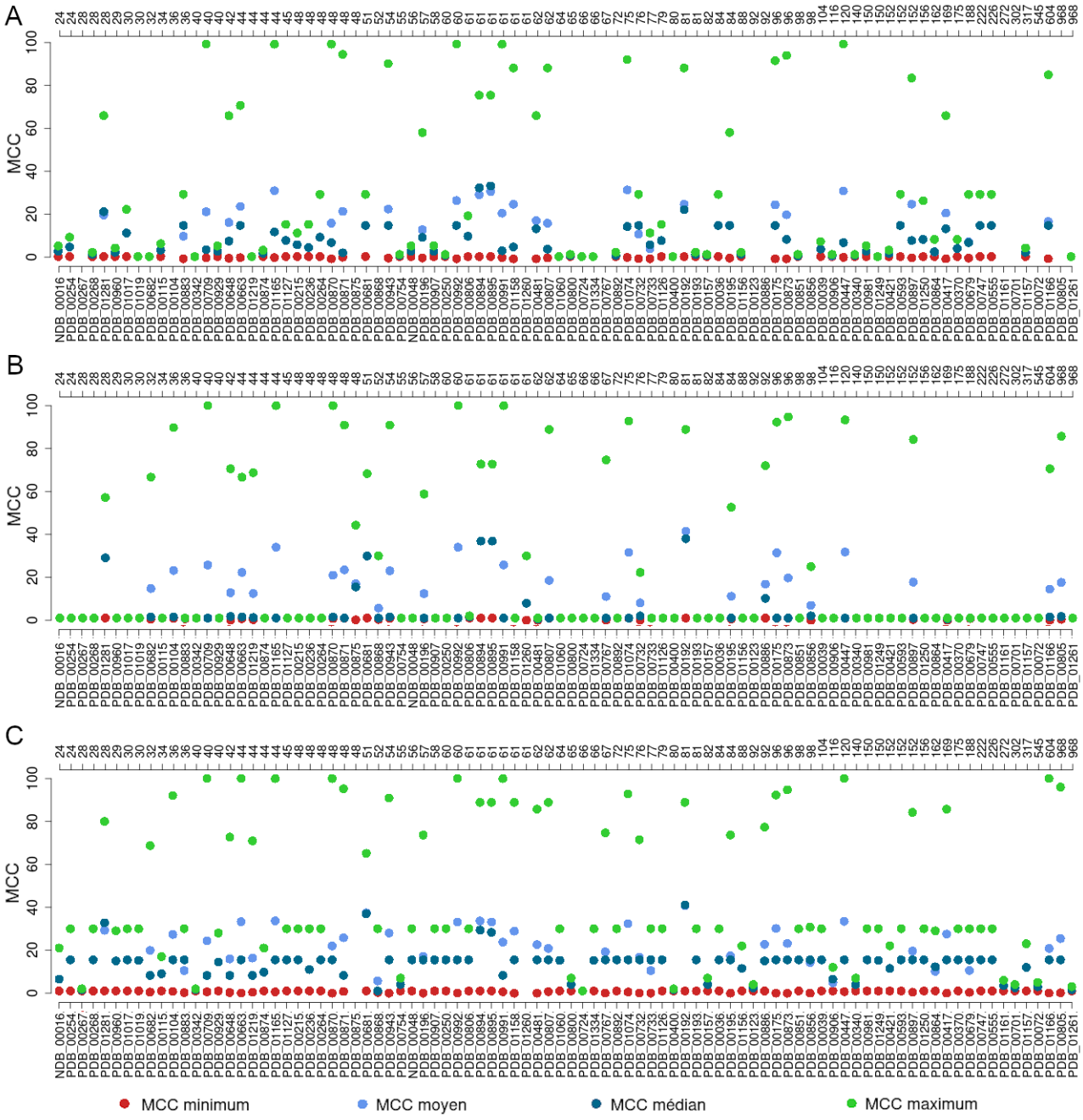

Figure 3.8 : MCC des structures d'entrée de RCPred pour chaque complexe de C_{multi} suivant l'outil utilisé. Moyenne, médiane, minimum et maximum MCC sur l'ensemble des structures secondaires d'entrées de RCPred générées soit par BiokoP (A), pKiss (B) et RNAsubopt (C). La longueur des complexes est indiquée par l'axe des ordonnées supérieur.

Comme nous l'avons évoqué précédemment, les structures utilisées en entrée sont de qualité assez faible, tandis que les interactions sont de qualité moyenne à bonne. Les résultats de RCPred sont meilleurs pour les complexes de tailles inférieures à environ 100 nucléotides, alors que quelque soit la longueur des ARN et des paires d'ARN en entrée, la qualité des structures et interactions utilisées en entrée semble similaire. Cela suggère que le modèle énergétique de

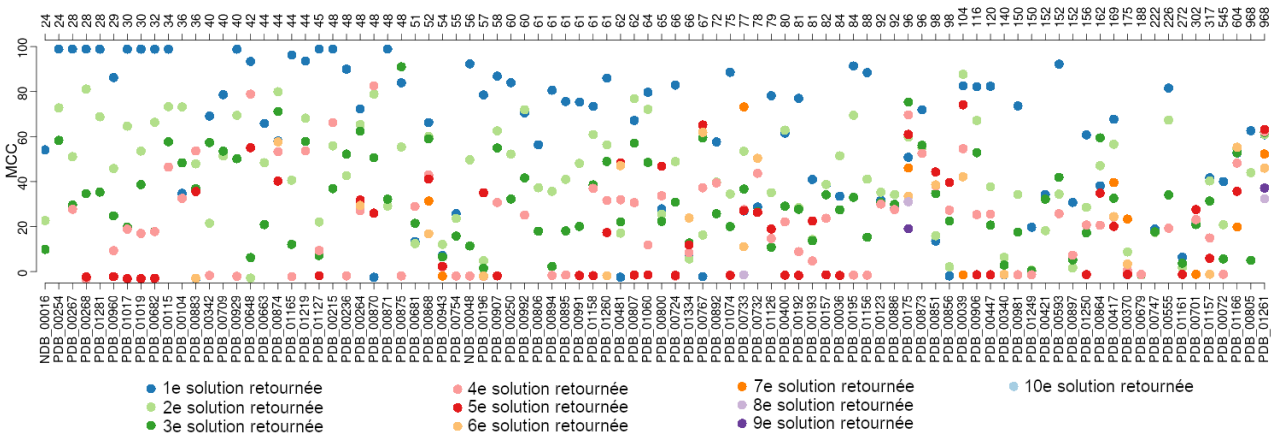

Figure 3.9 : MCC moyens sur 10 exécutions des 10 premières structures secondaires retournées par RCPred sur le jeu de données C_{multi}. Les structures secondaires utilisées en entrée sont générées conjointement par BiokoP, pKiss et RNAsubopt. La longueur des complexes est indiquée sur l'axe des ordonnées supérieur.

RCPred est performant pour les complexes de tailles inférieures à environ 100 nucléotides et qu'il est perfectible pour les complexes plus grands.

Une explication possible pour les complexes de grandes tailles est que la simple somme des énergies des structures et interactions composant un complexe introduit un biais et que ce biais augmente avec la longueur des ARN car ils ont une plus grande énergie. Une autre possibilité est que pour les ARN de grande taille, les outils de prédiction utilisés pour les entrées de RCPred proposent beaucoup plus de structures internes, rendant plus difficile le choix d'une bonne solution pour RCPred.

Influence des entrées

RCPred dépend d'un ensemble de structures secondaires et d'interactions déjà prédites. Nous étudions ici l'impact de ces entrées sur les résultats de RCPred en comparant les résultats obtenus indépendamment par BiokoP, pKiss et RNAsubopt. Les MCC moyens (sur 10 itérations) des 10 premières prédictions obtenues pour chaque complexe de C_{multi} sont présentés en figure 3.10 : les résultats obtenus avec en entrée les résultats de BiokoP (A), pKiss (B) et RNAsubopt (C). Nous pouvons tirer les même conclusions que précédemment pour chaque condition, à savoir :

— la qualité des solutions augmente quand l'énergie libre diminue,
— il est important de générer plusieurs solutions,
— les résultats sont meilleurs pour des petits complexes.

La comparaison des résultats obtenus avec les trois outils montre que la qualité des solutions est similaire pour les complexes de longueur inférieure à environ 100 nucléotides. En revanche, ce n'est pas le cas pour les autres complexes. Nous pouvons voir par exemple que pour BiokoP, il n'y a pas de résultats pour les grands complexes car certains ARN les composant ont une longueur trop importante pour que des structures soient prédites avec l'outil

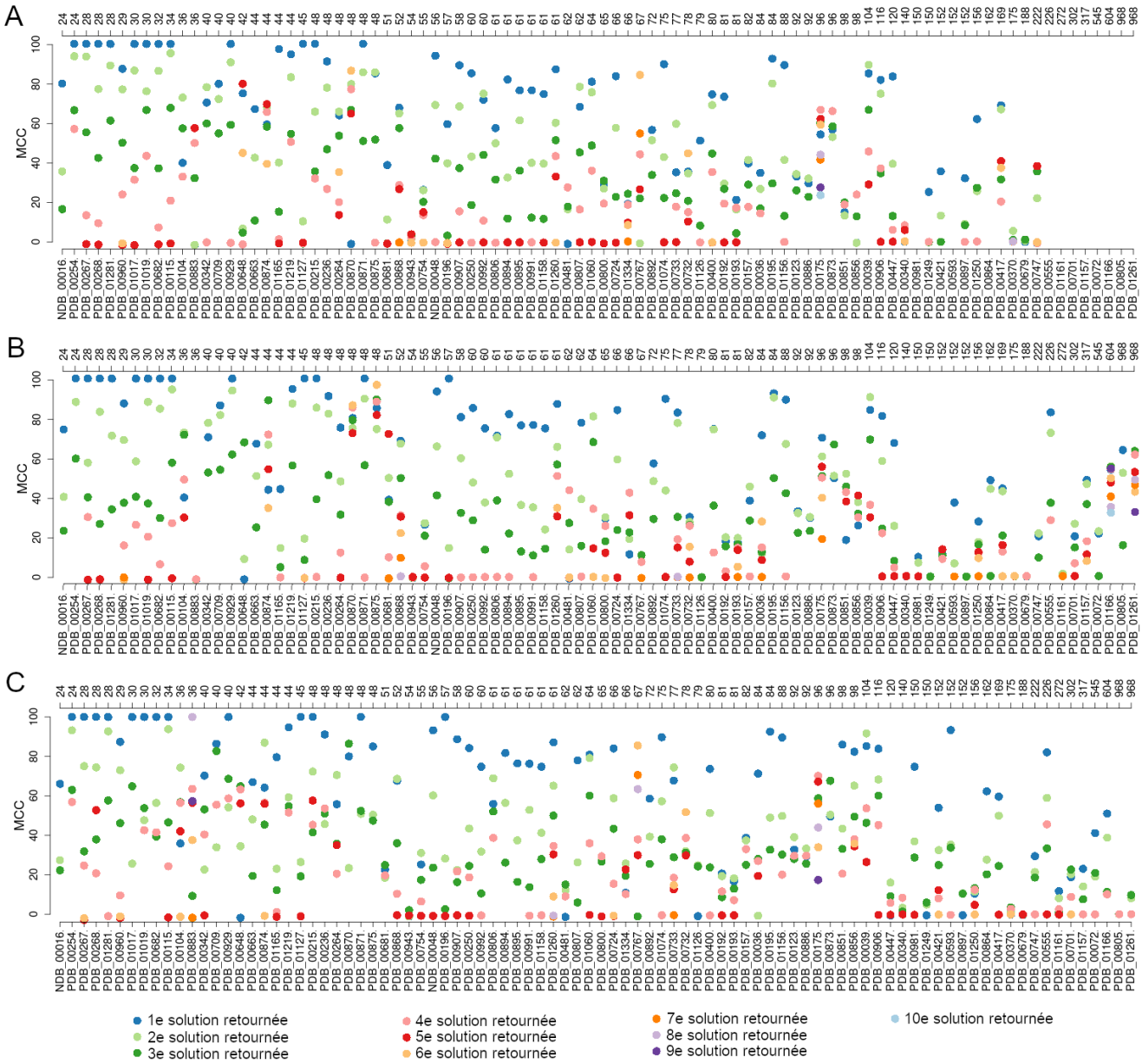

Figure 3.10 : MCC moyens sur 10 itérations des 10 premières structures secondaires retournées par RCPred sur le jeu de données C_{multi} en fonction des structures secondaires utilisées en entrée. Les structures secondaires utilisées en entrée sont générées soit par BiokoP (A), pKiss (B) ou RNAsubopt (C). La longueur des complexes est indiquée sur l'axe des ordonnées supérieur.

BiokoP. Pour ces complexes, les résultats de BiokoP, pKiss et RNAsubopt sont complémentaires, nous pouvons l'observer en figure 3.9, où l'alliance des entrées des trois outils permet d'obtenir de meilleurs résultats. Cela suggère que lorsque les entrées sont nombreuses, la diversité des structures et interactions est plus grande, ce qui permet à RCPred de trouver les meilleures structures et interactions pour former les structures de complexes. Cela confirme l'intérêt de l'approche modulaire de RCPred qui permet de traiter un grand nombre de structures secondaires et d'interactions en entrées, afin de prédire efficacement

et rapidement des complexes d'ARN.

3.3.5 Comparaison de RCPred avec la littérature

Nous avons comparé RCPred à tous les outils disponibles de la littérature, à savoir, NanoFolder [23], NUPACK [248] et MultiRNAfold [10]. Parmi ces outils, NanoFolder et RCPred peuvent générer des structures avec pseudonœuds. NUPACK et RCPred peuvent générer des solutions sous-optimales.

Les MCC obtenus avec ces outils sont présentés en figure 3.11. Les résultats de RCPred correspondent aux MCC moyens obtenus par la structure secondaire d'énergie minimale (A) de chaque complexe de C_{multi} sur 10 exécutions. Les résultats de NUPACK correspondent aux MCC obtenus par la structure secondaire d'énergie minimale (A) de chaque complexe du jeu de données. En figure 3.11 B il s'agit, pour RCPred et NUPACK, des MCC moyens obtenus par les structures secondaires les plus proches des structures réelles. Les résultats de NanoFolder et MultiRNAfold sont les MCC obtenus par la prédiction de chaque complexe du jeu de données C_{multi}. Le tableau 3.3 rassemble les moyennes sur le jeu de données. Nous n'avons pas pu tester NanoFolder, qui est disponible via un serveur web, sur les plus grands complexes (plus de 550 nucléotides) à cause de la limite de taille. De même, l'outil MultiRNAfold n'a pas pu être exécuté sur certains complexes. Pour ces deux outils, les moyennes ont été réalisées seulement sur les complexes dont la prédiction a pu être réalisée.

RCPred obtient de meilleurs résultats que tous les autres outils pour 35.5% des complexes. En deuxième position, on retrouve NanoFolder qui obtient de meilleurs résultats que tous les autres outils pour 23.3% des complexes, puis MultiRNAfold pour 11.1% et enfin NUPACK pour 8.8%. Pour les 21.1% de complexes restants, plusieurs outils obtiennent des résultats similaires.

Nous pouvons également observer, comme précédemment, que les résultats obtenus pour les complexes les plus longs sont moins bons avec RCPred, comme avec les autres outils. Les autres outils reposent aussi sur des modèles énergétiques qui peuvent être simples comme celui de NanoFolder et MultiRNAfold ou plus sophistiqués comme celui de NUPACK. Le fait est que les modèles thermodynamiques montrent des limites déjà dans la prédiction de structure d'un ARN seul : à partir d'une certaine longueur, la diversité des structures possibles est très importante et la moindre imprécision dans les paramètres énergétiques peut résulter en une mauvaise prédiction. D'autre part, certaines structures de références ne peuvent être prédites car les modèles de calculs de l'énergie sont des approximations :

— le modèle de calcul de l'énergie des pseudonœuds de RCPred ;
— l'énergie dans NanoFolder ne prend en compte que les empilements d'appariements ;
— l'énergie dans MultiRNAfold ne différencie pas les appariements internes et externes.

Rappelons aussi que MultiRNAfold et NUPACK ne prennent pas en compte dans leur modèle les pseudonœuds. Concernant RCPred, ses prédictions dépendent directement des structures et des interactions données en entrée. Ainsi, si les structures secondaires et interactions de références ne se trouvent

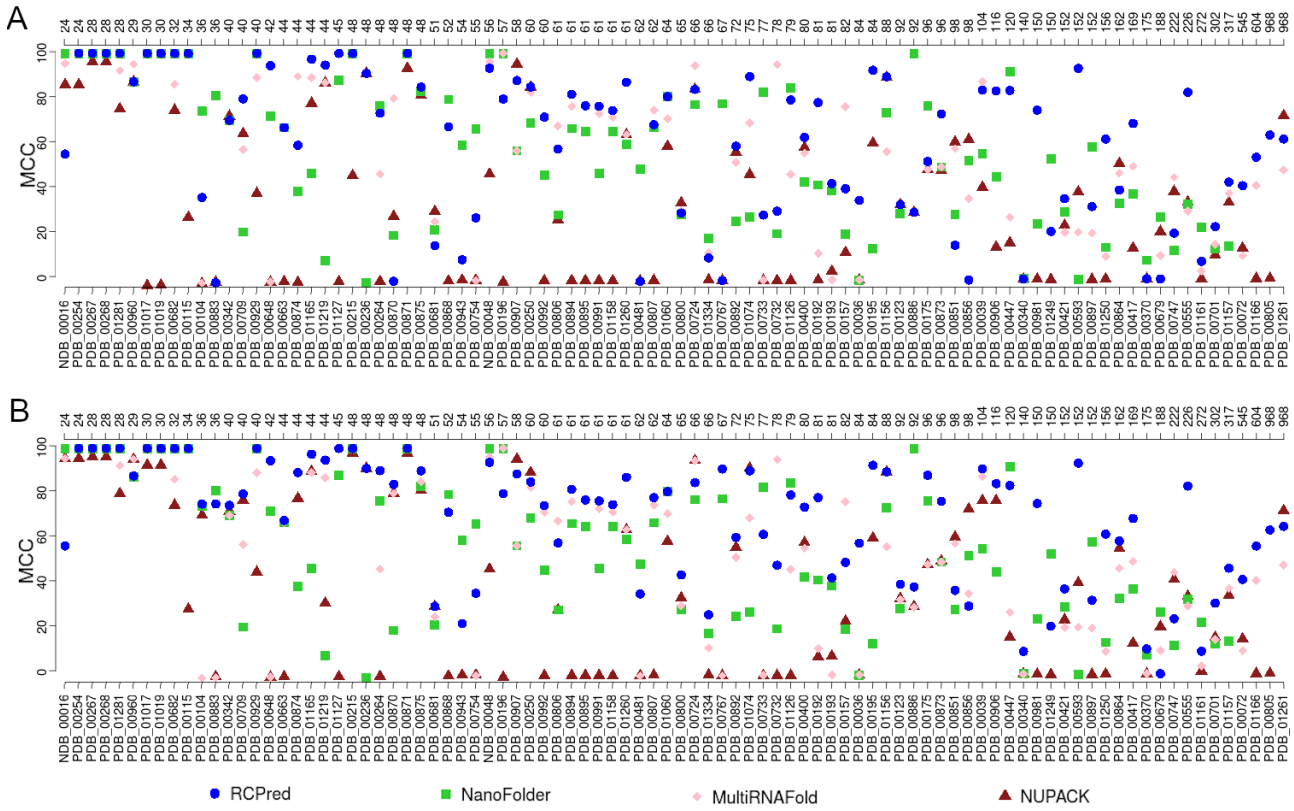

Figure 3.11 : MCC obtenus par RCPred, NanoFolder, NUPACK et MultiRNAfold sur chaque complexe de C_{multi}. La longueur des complexes est indiquée sur l'axe des ordonnées supérieur. A) RCPred : MCC moyens sur 10 itérations obtenus par les structures secondaires d'énergie minimale. NUPACK : MCC obtenus par les structures secondaires d'énergie minimale. NanoFolder et MultiRNAfold : MCC obtenus par les structures secondaires d'énergie minimale. B) RCPred : MCC moyens sur 10 itérations obtenus par les structures secondaires les plus proches de la structure de référence. NUPACK : MCC obtenus par les structures secondaires les plus proches de la structure de référence. NanoFolder et MultiRNAfold : MCC obtenus par les structures secondaires d'énergie minimale.

pas parmi ses données, les prédictions de RCPred ne peuvent être qu'éloignées des structures réelles (voir section 3.3.3).

RCPred obtient également de meilleurs résultats en moyenne, avec un MCC égal à 60.5% et supérieur de 10 points environ aux MCC de NanoFolder et MultiRNAfold, et de 30 points à celui de NUPACK. Les résultats sont similaires concernant les F_1-scores. RCPred a un PPV moyen (61.2%) supérieur à celui de tous les autres outils et une sensitivité égale à 61.2% similaire à celle de NanoFolder (61.9%) et supérieure à celle de NUPACK et MultiRNAfold (voir tableau 3.3). RCPred obtient les meilleurs résultats en moyenne et a un meilleure équilibre entre le nombre de prédictions d'appariements faux positifs et de faux négatifs, comparé aux autres outils.

La figure 3.11 B apporte des résultats complémentaires. Nous pouvons voir que grâce à l'ensemble des structures générées, nous pouvons garantir qu'au

	RCPred	NanoFolder	NUPACK	MultiRNAFold
Sensitivité	61.2	**61.9**	30.6	54.6
PPV	**61.2**	50.4	34.0	56.0
MCC	**60.5**	55.0	31.3	54.6
F_1score	**60.5**	54.9	31.73	54.8

Tableau 3.3 : Sensitivité, PPV, MCC et F_1-score moyens sur C_{multi} obtenus par RCPred, NanoFolder, NUPACK et MultiRNAFold. RCPred : statistiques moyennes sur 10 itérations obtenus par les structures d'énergie minimale. NUPACK, NanoFolder et MultiRNAfold : statistiques obtenues par les structures d'énergie minimale. En gras est indiquée la meilleure statistique.

moins une des structures prédites est proche de la structure de référence.

3.4 Discussion

Nous proposons une nouvelle méthode et un outil, appelé RCPred, pour prédire les structures secondaires de complexes d'ARN (composés de plus de deux ARN). Nous avons modélisé le problème comme un problème d'optimisation combinatoire, plus particulièrement en une recherche de clique de poids maximum dans un graphe. Le graphe est construit à partir de structures secondaires d'ARN et de sites d'interactions donnés par l'utilisateurs, et dans ce graphe, une clique correspond à une structure secondaire d'un complexe. Ce problème de clique est résolu grâce à une heuristique basée sur la *Breakout Local Search* (BLS) et implémentée dans notre outil RCPred.

Cette modélisation nous permet de prédire tous les types de motifs trouvés dans les complexes d'ARN, ainsi que les pseudonœuds internes et externes. Ces derniers ne sont pas prédits par deux autres outils prédisant des structures de complexes d'ARN, appelés NUPACK et MultiRNAfold. En revanche l'outil NanoFolder est capable de les prédire.

RCPred peut également prédire un ensemble de structures sous-optimales. Générer des structures sous-optimales est très important dans la prédiction de structures secondaires d'ARN pour plusieurs raisons. Tout d'abord, il est admis que la structure réelle d'un ARN n'est pas toujours la structure d'énergie libre minimale, mais souvent une structure proche. De plus, un modèle ne peut capturer toutes les subtilités de la thermodynamique. Les structures sous-optimales permettent alors de prédire des structures qui sont plus proches de la structure réelle. Parmi les outils de l'état de l'art, seul l'outil de l'ensemble logiciel NUPACK permet de générer des structures sous-optimales.

Nous avons testé RCPred sur un jeu de données de 90 complexes d'ARN de longueurs variées et pouvant inclure des pseudonœuds. Nous avons montré que RCPred est capable de prédire des structures secondaires de complexes plus précisément que les outils disponibles dans la littérature, à savoir NUPACK, MultiRNAfold et NanoFolder. Nous avons également pu voir que les prédictions étaient meilleures pour les complexes de longueurs inférieures à environ 100 nucléotides. Le modèle énergétique pourrait en être la cause. Dans le cas de

RCPred, l'énergie libre d'un complexe est calculée à partir de la somme des structures et des sites d'interaction le composant. Une perspective pourrait être d'améliorer le calcul de l'énergie libre en recalculant les énergies des complexes trouvés par RCPred en intégrant les coûts entropiques liés à la formation des pseudonœuds et des interactions. Les complexes seraient ensuite triés selon les nouvelles énergies. Le calcul de l'énergie libre d'un complexe pourrait être réalisé en adaptant la méthode de calcul de l'outil RNAeval de ViennaRNA aux complexes de plus de deux ARN.

Les résultats de RCPred dépendent des structures et interactions données en entrée de celui-ci. Nous avons montré à l'aide de trois outils, BiokoP, pKiss et RNAsubopt, que l'union de leur prédiction de structures secondaires permettait à RCPred d'obtenir de meilleurs résultats, c'est-à-dire que la diversité apportées par un ensemble d'outils permet d'avoir une plus grande probabilité d'obtenir des structures secondaires de référence. Il faut donc favoriser un grand nombre de structures et interactions en entrée. Cela montre également que notre outil, RCPred, est capable de sélectionner les meilleures structures et interactions, et ce parmi un grand nombre, pour former des structures de complexes d'ARN.

En analysant les résultats donnés par RCPred nous avons pu constater quelques limites. Tout d'abord, certaines des structures sont très similaires, une perspective serait de regrouper ces structures et de proposer à l'utilisateur un représentant pour chaque groupe.

Ensuite, nous avons réalisé que quand des pseudonœuds sont prédits dans les structures de complexes, ceux-ci pouvait avoir une profondeur élevée. Pour rappel, la profondeur est le nombre sous-structures sans pseudonœuds dans laquelle peut être décomposée une structure secondaire (voir définition en section 2.2.1). Nous avons fait le même constat pour l'outil NanoFolder. Or, parmi les structures validées expérimentalement trouvées dans les bases de données, la profondeur d'un pseudonœud excède rarement 2. Une amélioration à prévoir pour augmenter la performance de prédiction est donc de contrôler la profondeur des pseudonœuds prédits. Cela sera étudié dans le chapitre suivant.

4 Prédiction interactive de structures secondaires de complexes d'ARN

Dans ce chapitre, nous proposons une nouvelle méthode pour prédire, de manière interactive, des structures secondaires de complexes d'ARN. En effet, l'utilisateur biologiste possède des informations qui peuvent aider à la prédiction de structures. Cela peut être des informations sporadiques sur la structures (appariements, motifs), pour lesquelles nous avons mis en place des *contraintes utilisateurs*. Il peut également être en possession de données structurales telles que le SHAPE, le DMS ou encore le PARS (voir section 1.1.6, page 18). L'intégration de ces contraintes utilisateurs et données structurales dans la prédiction de structures secondaires existe pour les ARN [189] mais pas pour les complexes d'ARN. Nous souhaitons donc proposer une nouvelle méthode et un nouvel outil les intégrant. Le nouvel outil est appelé C-RCPred pour *Constrained RNA Complex Prediction*.

Pour cela, nous nous basons sur la méthode développée au chapitre précédent qui prend en entrée un ensemble de structures secondaires par séquence d'ARN et un ensemble d'interactions ARN-ARN par paire d'ARN. Cette méthode, implémentée dans l'outil RCPred, vise à trouver les combinaisons de ces entrées, ayant les plus petites énergies libres. Cette méthode consiste à résoudre de manière approchée avec une heuristique un problème d'optimisation mono-objectif, le problème de la clique maximum, où l'énergie libre est assimilée à la fonction objectif. La nouvelle version que nous proposons est, quant à elle, multi-objectif. Les nouveaux objectifs de prédiction correspondent au respect de contraintes utilisateurs et à l'accord avec des données structurales.

Dans la nouvelle méthode que nous proposons, nous voulons générer un ensemble de Pareto approché (voir section 1.2.3, page 27), permettant d'obtenir des structures de complexes optimisant les différents objectifs. Or, pour le problème de la clique maximum qui est NP-difficile, il n'existe pas d'heuristique permettant de générer l'ensemble de Pareto approché. Nous proposons ici une heuristique multi-objectif, basée sur la recherche locale et la recherche tabou, pour calculer l'ensemble de Pareto approché. Contrairement à la méthode du chapitre précédent, nous ne cherchons pas à obtenir des structures secondaires

sous-optimales. En effet, avec notre heuristique nous n'avons pas de garantie de performance et l'ensemble de Pareto approché obtenu peut être éloigné de l'ensemble de Pareto exact. Ainsi, si nous cherchions à générer des ensembles de Pareto sous-optimaux approchés, ils pourraient être très éloignés des ensembles de Pareto sous-optimaux exacts. C'est pour cela que nous choisissons, d'approcher uniquement le premier ensemble de Pareto exact.

Dans notre nouvelle méthode, nous mettons en œuvre des solutions pour répondre à des limites de RCPred. Tout d'abord, nous savons que lorsqu'un complexe d'ARN se forme, quelques modifications peuvent avoir lieu dans les structures des ARN pour permettre les interactions. Dans notre nouvelle méthode, nous permettons d'autoriser des conflits, grâce à un seuil donné par l'utilisateur, pour modéliser ce phénomène.

Une autre limite de RCPred est qu'il ne permet pas de limiter la profondeur des pseudonœuds (voir section 1.1.4, page 12). Or, les pseudonœuds référencés dans les bases de données ont en général une profondeur égale à 2 ou 3. En effet, dans l'ancienne version, la profondeur n'est pas régulée et peut donc avoir une valeur quelconque dans les structures prédites. Dans ce chapitre, la profondeur des pseudonœuds est déterminée en trouvant le nombre chromatique de graphes de cercle, similaires aux graphes d'intervalles (voir les définitions en section 1.2.2, page 24).

Par ailleurs, nous nous intéressons à la classification des structures secondaires de complexes d'ARN. En effet, notre nouvelle méthode, tout comme RCPred, retourne plusieurs structures, mais celles-ci peuvent être très proches les unes des autres. Il n'existe pas d'outil disponible permettant directement de classer les structures secondaires de complexes d'ARN. Nous proposons donc ici une nouvelle méthode basée sur la classification spectrale à noyau pour classer les structures que nous prédisons.

Toutes ces nouvelles fonctionnalités ont donné lieu à l'outil C-RCPred, dont le principe est illustré en figure 4.1. C-RCPred prend en entrée des structures secondaires et des interactions, ainsi que des contraintes utilisateurs et des données structurales. À partir de ces entrées, C-RCPred prédit un ensemble de complexes répondant aux différents critères. Les structures de ces complexes sont visualisées grâce à une interface graphique dynamique, à partir de laquelle l'utilisateur peut ajouter de nouvelles contraintes. Les nouvelles contraintes sont utilisées pour relancer l'outil C-RCPred afin d'effectuer de nouvelles prédictions. Ce cycle peut être répété autant de fois que nécessaire.

Ce chapitre débute par un état de l'art des méthodes de prédiction de structures d'ARN intégrant des contraintes utilisateurs et des données structurales, ainsi que des méthodes de classification de structures secondaires d'ARN. Nous décrivons ensuite notre méthode multi-objectif, comment nous autorisons des conflits entre nucléotides et comment nous limitons la profondeur des pseudonœuds. Nous présentons également notre programme linéaire multi-objectif modélisant le problème de la clique maximum, avec les deux nouvelles fonctions objectifs, et nous présentons une heuristique tri-critère pour le résoudre. Nous décrivons ensuite notre méthode de classification de structures secondaires de complexes d'ARN. Puis, nous présentons les résultats des premiers tests que nous avons effectués avec notre outil C-RCPred pour

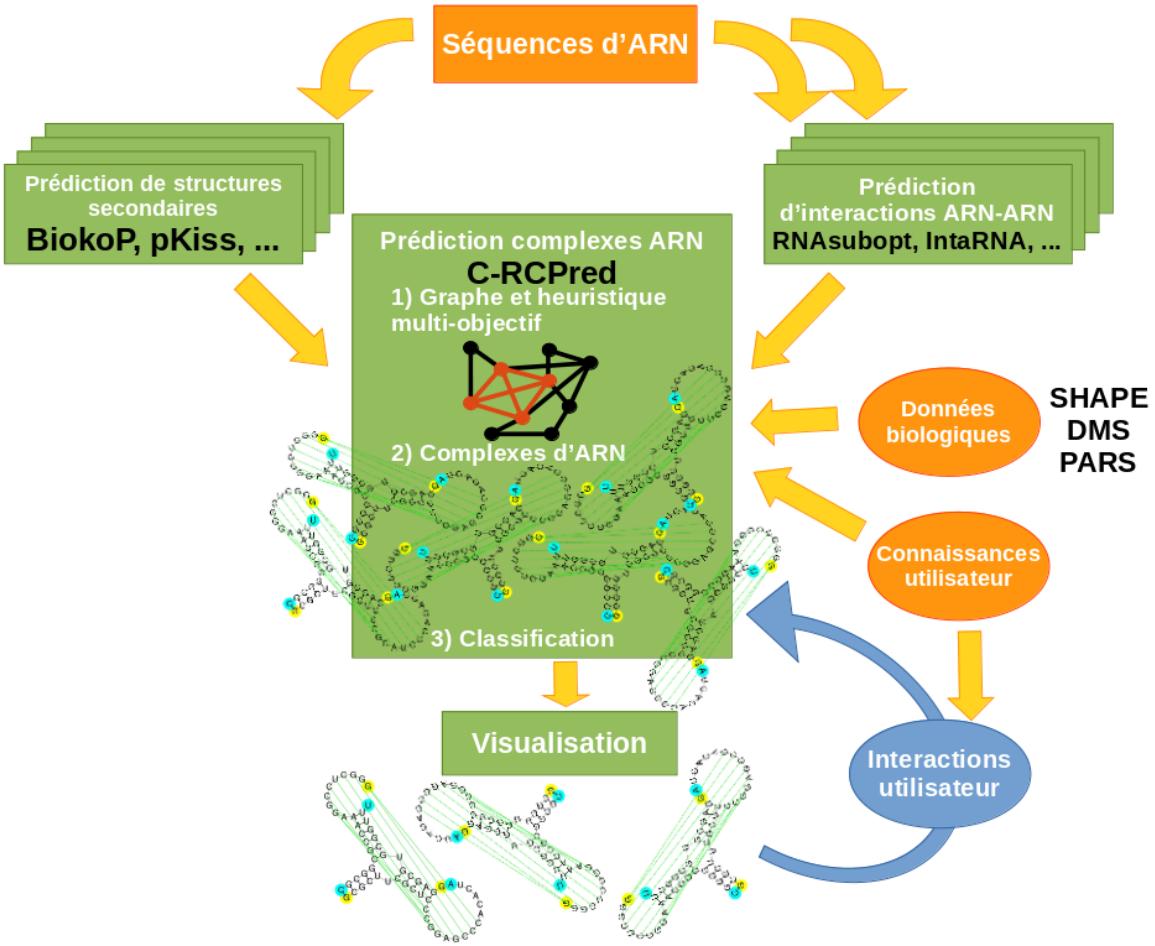

Figure 4.1 : Principe de l'outil C-RCPred.

évaluer l'efficacité de notre méthode. Enfin, la dernière section est dédiée à la discussion des résultats.

4.1 État de l'art

L'interactivité dans la prédiction de structures d'ARN est réalisée grâce à la prise en compte de contraintes dans les algorithmes. Ces contraintes sont dites *fortes* (voir définition 1.2.13) si elles doivent être absolument respectées, ce sont par exemple des contraintes sur les appariements, non-appariements, longueurs, présence ou non de pseudonœuds. Ces contraintes font que certaines structures sont complètement ignorées. Au contraire, des contraintes sont dites *faibles* si elles orientent simplement les résultats, c'est le cas la plupart du temps pour la prise en compte de données structurales telles que le SHAPE. Ces contraintes favorisent ou défavorisent certaines structures.

Dans cet état de l'art, nous présenterons dans un premier temps, les méthodes et outils prenant en compte des contraintes fortes, puis ceux prenant en compte des contraintes faibles. Nous étudierons également les méthodes visant à classer les structures secondaires d'ARN et de complexes d'ARN.

4.1.1 Interactivité via des contraintes fortes

Nous présentons ici des méthodes et outils prédisant des structures secondaires d'ARN prenant en compte des contraintes fortes. Un récapitulatif des outils présentés est donné dans le tableau 4.1.

Référence	Modèle	Outil	Structures sous-optimales	Pseudonœuds
Zuker et Stiegler, 1981 [259]	MFE	Mfold	✓	
Cedergren et al., 1988 [36]	MFE	RNASE		
Gaspin et Westhof, 1995 [78]	MFE	SASSPARN	✓	✓
Reeder et Giegerich, 2004 [176]	MFE	pknotRG	✓	
Giegerich et al., 2004 [80]	MFE	RNAshapes	✓	
Reeder et al., 2007 [175]	MFE	Locomotif	✓	
Parisien et Major, 2008 [164]	MFE	MC-Fold	✓	✓
Markham et Zuker, 2008 [136]	MFE	UNAFold	✓	
Engelen et Tahi, 2010 [72]	MFE	Tfold	✓	✓
Reuter et Mathews, 2010 [183]	MFE	RNAstructure	✓	
Lorenz et al., 2011 [130]	MFE	RNAfold	✓	
Janssen et Giegerich, 2014 [101]	MFE	pKiss	✓	✓

Tableau 4.1 : Récapitulatif des outils de prédiction de structures secondaires prenant en compte des contraintes fortes. Les outils en gris ne sont plus ou pas encore disponible, ou ils sont obsolètes.

La possibilité de prendre en compte des contraintes dans les outils de prédiction est considérée dès les années 80. Un outil très connu, appelé Mfold [259, 256], permet par exemple de prendre en amont des contraintes qui modifient grandement les valeurs de l'énergie libre de son algorithme. L'algorithme est un algorithme récursif de minimisation de l'énergie libre, les modifications des valeurs de l'énergie libre vont modifier les récursions effectuées par l'algorithme. Il est maintenant remplacé par l'outil UNAFold [136]. Les contraintes permettent de forcer ou d'interdire des appariements, ou des nucléotides non-appariés. D'autres programmes [237, 246] permettent de calculer un ensemble de structures prenant en compte des données biochimiques, comme des modifications sur les nucléotides.

D'autres méthodes, plus interactives sont ensuite développées. Par exemple, le programme RNASE [36], permet à l'utilisateur de sélectionner des hélices préalablement calculées, pour orienter les prédictions. Dans les années 90, Gaspin et Westhof développent un outil appelé SAPSSARN (Système d'Aide à la Prédiction de Structures Secondaires d'ARN) [78] qui calcule toutes les structures sous-optimales (incluant les pseudonœuds) satisfaisant des contraintes utilisateurs. Le programme permet de rajouter ou d'enlever des contraintes à tout moment.

Plus tard, des programmes appelés *thermodynamic matchers* sont développés [176, 175]. Ces outils consistent à générer un ensemble de structures correspondant à des motifs. Par exemple, dans les outils appelés pknotRG [176] et Locomotif [175], les contraintes sont définies à partir d'une grammaire, ap-

pelée grammaire de programmation dynamique algébrique, qui permet de générer des structures secondaires selon des motifs. Dans la même idée que les *thermodynamic matchers*, RNAshapes [80, 177] utilise des formes abstraites pour représenter les structures secondaires et les rassembler. Les contraintes peuvent être prises en compte grâce à ces formes abstraites qui permettent le calcul rapide de structures secondaires. Un outil, appelé pKiss [101], peut également faire usage des formes abstraites comme contraintes fortes, ainsi que des contraintes permettant la prédiction d'un certain type de pseudo-nœuds (type H, type HHH, ou type H et HHH ensemble) ou de ne pas prédire de pseudonœuds.

L'ensemble logiciel RNAstructure [183] contient des outils prenant en compte des contraintes fortes. Les contraintes permettent de forcer ou d'interdire des appariements, ou des nucléotides non-appariés. Dans ces outils, les contraintes sont utilisées pour pénaliser des régions : de très grandes énergies sont données aux régions qui ne sont pas cohérentes avec les contraintes [140]. Cette méthode est aussi utilisée par l'outil appelé MC-Fold [164].

L'ensemble logiciel ViennaRNA [130] contient également des outils prenant en compte des contraintes fortes. Parmi ces outils, RNAfold utilise un algorithme classique de minimisation d'énergie et prend en compte les contraintes lors des récursions de l'algorithme. Pour les contraintes d'appariements, il suffit de ne pas effectuer les récursions concernées. Les contraintes fortes des outils de ViennaRNA assurent que les structures prédites respecteront les contraintes mais les contraintes ne sont pas obligatoirement présentes dans les structures. Par exemple si la contrainte est un appariement (i, j), les structures prédites n'auront pas d'appariements (x, y) tels que $i < x < j < y$, $x < i < y < j$ ou $|\{x, y\} \cap \{i, j\}|$.

Au laboratoire IBISC, un outil de prédiction de structure secondaire, appelé Tfold [72], a été développé. Il utilise l'approche comparative et est capable de prendre en compte des contraintes. Ces contraintes permettent d'indiquer la position d'hélices.

Dans les outils de prédiction d'interactions ARN-ARN, des contraintes fortes sont parfois prises en considération. Par exemple, un outil appelé IntaRNA [31] permet d'indiquer une zone pour un ARN, où le site d'interaction est susceptible de se trouver.

4.1.2 Interactivité via des contraintes faibles

Cette section est dédiée aux méthodes et outils prédisant des structures secondaires d'ARN prenant en compte des contraintes faibles. La plupart des ces outils prennent en compte des données structurales, telles que le SHAPE (voir section 1.1.6, page 18). Les outils prenant en compte des données structurales sont récapitulés dans le tableau 4.2.

Des outils de l'ensemble logiciel RNAstructure sont capables de prendre en compte les données de SHAPE [54, 132]. Ces outils utilisent uniquement les données correspondant à des nucléotides appariés, puis les transforment en pseudo-énergies qui serviront à la prédiction classique. La pseudo-énergie associée à un nucléotide i en fonction de sa réactivité au SHAPE est définie

Référence	Modèle	Outil	Données supportées	Structures sous-optimales	Pseudo-nœuds
Reuter et Mathews, 2010 [183]	MFE	Fold	SHAPE	✓	
Zarringhalam et al., 2012 [249]	MFE	RNAsc	SHAPE	✓	
Washietl et al., 2012 [230]	MFE	RNApbfold	SHAPE	✓	
Hajdin et al., 2013 [86]	MFE	ShapeKnots	SHAPE	✓	✓
Ouyang et al., 2013 [160]	Échantillonnage	SeqFold	SHAPE, PARS, FragSeq		
Wu et al., 2015 [240]	MEA	RME	SHAPE, PARS, DMS	✓	
Spasic et al., 2017 [204]	Échantillonnage	Rsample	SHAPE	✓	

Tableau 4.2 : Récapitulatif des outils de prédiction de structures secondaires prenant en compte des données structurales.

comme suit :

$$\Delta G(i) = m \times ln(\text{Réactivité SHAPE}(i) + 1) + b. \tag{4.1}$$

La variable m est la pente de la fonction, elle est négative et signifie que l'énergie $G(i)$ augmente lorsqu'un nucléotide apparié a une réactivité au SHAPE basse. La variable b est l'ordonnée à l'origine de la fonction, elle est positive et pénalise l'énergie des nucléotides appariés ayant une réactivité haute. Cette fonction a été utilisée car la réactivité est inversement proportionnelle à la probabilité d'appariement d'un nucléotide. Elle est très utilisée dans les outils prenant en compte des données de SHAPE.

En 2010, une étude [173] s'est concentrée sur l'influence de la qualité des données (SHAPE, DMS ou nucléases) sur les prédictions de structures secondaires. Les auteurs de cette étude ont souhaité déterminer quel type de données donne le plus d'informations structurales et quelle est la proportion de bruit dans chaque type de données. Le principe de leur méthode est d'échantillonner des structures d'ARN avec l'outil Sfold [59] et de choisir la structure qui correspond le mieux aux données structurales. Ils ont étudié différentes métriques (sensitivité, PPV et distance de Manhanttan) pour évaluer les structures en fonction des données structurales et ont ainsi montré que lorsque les données contiennent peu de bruit, ils sont capables de prédire exactement la structure d'un ARN.

Après cette étude, plusieurs méthodes sont développées pour essayer de minimiser l'effet du bruit dans les données structurales. C'est le cas de l'outil RNApbfold de ViennaRNA [230] qui traduit les données structurales en pseudo-énergies. Contrairement aux autres algorithmes utilisant des pseudo-énergies, leurs valeurs d'énergie sont modifiées seulement si cela est nécessaire, c'est-à-dire, seulement si les probabilités d'appariements prédites avec les données structurales sont en accord avec les probabilités d'appariements prédites sans les données structurales. Cela permet de s'assurer de la justesse des données dans une certaine mesure.

Un outil, appelé RNAsc [249], cherche lui aussi à s'assurer de la différence entre les probabilités d'appariements, calculées avec ou sans données structurales. Pour cela, il calcule les pseudo-énergies et les probabilités de non-appariements associées à chaque nucléotide. Il cherche ensuite la structure secondaire qui minimise la différence entre les probabilités de non-appariements calculées avec ou sans données structurales.

Comme expliqué dans les chapitres précédent, nous savons que certains ARN peuvent avoir plusieurs structures. Il a été montré que les données de SHAPE d'un ARN, ayant plusieurs structures, peuvent correspondre à chacune de ses structures [233, 232]. Cela apporte une autre forme "d'imprécision" dans les données.

Spasic et al. développent en 2017 un outil, appelé Rsample (Restrained sample) [204], qui permet d'échantillonner les structures secondaires tout en prenant en compte le bruit dans les données. Pour cela, il cherche à minimiser la différence entre les données structurales et les données d'échantillonnage estimées par un modèle. La prédiction des structures est ensuite réalisée grâce au calcul de la fonction de partition contrainte. Les auteurs supposent que les données structurales de SHAPE sont une moyenne des réactivités qui sont pondérées par chaque structure que peut prendre un ARN. Pour estimer les réactivités de chaque nucléotide, les auteurs utilisent trois distributions aléatoires, à savoir la réactivité des nucléotides non-appariés, la réactivité des nucléotides impliqués dans des hélices terminales et la réactivité des nucléotides impliqués dans des hélices non terminales.

Une étude [24] a été réalisée par Bindewald et al. pour montrer quels modèles sont les plus à même de calculer les probabilités d'appariements grâce aux données de SHAPE, en prenant en compte les différentes structures possibles qu'un ARN peut avoir. Pour prédire des structures secondaires, les auteurs n'ont pas transformé les réactivités en pseudo-énergie comme la plupart des autres outils. Ils ont transformé les réactivités en probabilités d'appariements Watson-Crick grâce à des modèles appelés modèles probabilistes postérieurs Bayésiens. Avec un modèle postérieur Bayésien, la probabilité d'appariement d'une base i avec une base inconnue, notée $P(S_i = 1|D_i)$, est calculée à partir de la donnée structurale D_i, grâce à l'équation suivante (formule de Baye) :

$$P(S_i = 1|D_i) = \frac{P(D_i|S_i = 1) \times P(S_i = 1)}{P(D_i|S_i = 1)P(S_i = 1) + P(D_i|S_i = 0) \times P(S_i = 0)}. \tag{4.2}$$

$P(D|S)$ est la probabilité d'observer une certaine valeur expérimentale, notée D, étant donné que la base soit appariée ou non (S). Cette probabilité peut être estimée à partir de données structurales d'un ensemble d'ARN d'entraînement dont les structures sont connues. Les auteurs ont constaté que la réactivité du SHAPE était influencée par l'état des nucléotides voisins. En effet, ils ont montré que les modèles les plus efficaces sont ceux prenant en compte les nucléotides voisins.

Un outil, appelé RME (*Restrained Max-Expect*) [240], est basé sur un de ces modèles probabilistes postérieurs. RME est une amélioration de l'outil MaxExpect [134] qui prédit la structure secondaire d'un ARN en maximisant

la précision attendue (MEA). Les probabilités d'appariements calculées avec le modèle postérieur Bayésien sont utilisées pour restreindre le calcul de la fonction de partition. RME est capable de prendre en compte différent types de données : le SHAPE, le DMS et le PARS. Ce n'est pas le cas de la plupart des outils qui se concentrent essentiellement sur les données de type SHAPE. Or, comme nous avons pu le voir en section 1.1.6 (page 18), il existe de nombreuses autres données structurales. On notera, cependant, qu'un algorithme adapté aux données SHAPE (RNAstructure [54]) a été utilisé, avec succès, avec des données DMS [43]. De plus, un webserver a également été développé par les auteurs de l'outils RME afin de pouvoir utiliser différents types de données structurales avec plusieurs algorithmes de prédiction de structures (MaxExpect [134, 241], SeqFold [160], RNAstructure [140] et RNAfold [130]). Les auteurs mettent également à disposition des données structurales de plusieurs transcriptomes : humain, souris, levure et *Arabidopsis thaliana*.

Parmi tous les outils cités précédemment, aucun ne peut prédire des pseudonœuds et prendre en compte des données structurales. À notre connaissance, seuls deux outils en sont capables. Il s'agit des outils ShapeKnots [86] et MC-Fold [164]. Dans ces deux outils, les réactivités SHAPE sont transformées en pseudo-énergies avec l'équation 4.1. Pour intégrer les pseudonœuds dans le modèle d'énergie de ShapeKnots, des pénalités d'énergie sont ajoutées en fonction des distances entre les nucléotides impliqués dans le pseudonœud. Ces pénalités sont ajoutées après que les pseudo-énergies des nucléotides soient calculées. Dans l'outil MC-Fold, une fois les pseudo-énergies calculées, un algorithme classique de minimisation de l'énergie est utilisé.

Les outils cités précédemment sont tous dédiés à la prédiction de structures secondaires à l'échelle d'un seul ARN. Il existe quelques méthodes et outils destinés à prédire des structures secondaires à l'échelle génomique. Par exemple, un outil, appelé SeqFold [160], permet prédire des structures secondaires à l'échelle génomique. Pour cela, des données structurales sont transformées en un profil structural qui indique si un nucléotide est préférentiellement apparié ou non. Ce profil est utilisé pour échantillonner, de manière contrainte, des structures d'ARN de l'ensemble de Boltzmann. SeqFold peut utiliser des données de types PARS, SHAPE-Seq, FragSeq ou SHAPE.

Un outil de prédiction d'interactions ARN-ARN fait également usage de données SHAPE [148]. Pour cela, les données sont intégrées dans le calcul des probabilités d'accessibilité (probabilité de non-appariement d'un nucléotide) de chaque ARN. Ces probabilités sont ensuite utilisées pour prédire une interaction.

Un autre type de contraintes faibles a également été développé, ce sont les contraintes longues distances. Une étude [9] montre que les outils de thermodynamique prédisent de trop nombreux appariements longues distances (les nucléotides appariés sont très éloignés). La contrainte vise à diminuer l'énergie de ces appariements pour diminuer leur insertion dans les structures. Plusieurs outils disposent de ce type de contraintes : RNALFold (Vienna RNA [130]), Rfold [112], Raccess [113] ou encore CoFold [169]. Dans cette étude, les auteurs présentent un modèle MEA avec une pénalité sur les appariements longues distances permettant d'avoir des prédictions plus précises qu'avec un

modèle de MEA classique ou un algorithme de minimisation d'énergie modifié avec une fonction de pénalisation d'énergie (comme c'est le cas pour l'outil RNALFold). Une fonction de pénalité est utilisée dans le modèle MEA pour altérer le score MEA de chaque structure : plus la distance entre des nucléotides est longue, moins l'appariement de ces nucléotides est probable.

4.1.3 Classification de structures secondaires d'ARN

Nous abordons ici la classification des structures secondaires, d'un ARN seul ou d'un complexe. En effet, beaucoup d'outils prédisent des ensembles de structures secondaires et ces structures peuvent être très similaires, ce qui nécessite une classification. Les méthodes et outils présentés sont récapitulés dans le tableau 4.3.

Référence	Outil	Méthode	Complexe d'ARN
Shapiro and Zhang, 1990 [199]	RNAdistance	Distance	
Hochsmann et al., 2003 [92]	RNAforester	Distance	
Allali and Sagot, 2008 [7]	MiGaL	Distance	
Ouangraoua et al., 2007 [159]	Gardenia	Distance	
Herrbach, 2007 [90]	NestedAlign	Distance	
Guignon et al., 2008 [84]	RNAStrAT	Distance	
Ding et al., 2005 [60]	Sfold	Distance	
Freyhult et al., 2007 [74]		Distance	
Freyhult et al., 2007 [75]	RNAbor	Distance	
Liu et al., 2008 [127]	RNAcluster	Distance	
Agius et al., 2010 [3]	PR2S2Clust	Distance	
Lin et al., 2018 [126]		Distance	
Heyne et al., 2012 [91]	GraphClust	Noyau	
Miladi et al., 2017 [147]	RNAscClust	Noyau	
Janssen and Giegerich, 2014 [101]	RNAshapes	Distance	
Huang et al., 2012 [94]	RNAHelices	Distance	
Rogers and Heitsch, 2014 [185]	RNAprofiling	Distance	
Mneimneh and Ahmed, 2016 [150]		Distance	✓

Tableau 4.3 : Récapitulatif des méthodes et outils de classification de structures secondaires d'ARN et de complexe d'ARN.

Les premières classifications de structures sont réalisées grâce à des mesures de distance basées sur les appariements. Les premiers outils permettant de calculer la distance entre des structures secondaires, à savoir RNAdistance [199], RNAforester [92], MiGaL [7], TreeMatching [159], Gardenia [27], NestedAlign [90] et RNAStrAT [84], se basent sur différents modèles de structures secondaires, comme des arbres, des modèles multi-couches, ou encore, des séquences annotées par des arcs. Ils sont tous basés sur la distance utilisée par les premières approches d'alignement [199, 251, 104, 105].

En 2005, Ding et al. développent une méthode [60] générant l'ensemble de Boltzmann et réalisent une classification hiérarchique grâce à une distance basée sur les appariements (distance euclidienne) pour trouver les centroïdes. Cette méthode a été ensuite intégrée dans l'outil Sfold [59].

Peu après, Liu et al. [127] développent un outil, appelé RNACluster, qui permet de comparer et regrouper des structures secondaires. L'outil propose 6 mesures de distances : *base pair distance* (distance euclidienne), *mountain distance, morphological distance, tree edit distance, string edit distance et structural matrix distance*. La *mountain distance* et la *morphological distance* sont elles aussi basées sur les appariements, tandis que la *tree edit distance* est basée sur une représentation en arbre de la structure et *la string edit distance* sur l'alignement des strutures. La *structural matrix distance* utilise une matrice en 6 dimensions pour représenter les occurrences des différents appariements des structures. L'algorithme de classification utilisé est basé sur un arbre couvrant de poids minimal.

En 2007, un outil appelé RNAbor [75, 74] a été développé pour déterminer des structures secondaires voisines d'une structure donnée. RNAbor utilise une distance basée sur les appariements pour classer les structures voisines en fonction de la structure donnée.

En 2010, Agius et al. [3] développent une méthode de classification basée sur une distance relaxée des appariements et sur l'algorithme des k-moyennes. Leur méthode a été implémentée dans le but d'optimiser le temps d'exécution et est incorporée dans l'outil UNAfold [136].

Un outil, appelé PR2S2Clust [26], est dédié à la classification des *reads* issus de séquençage d'ARN et propose 7 mesures de distances basées sur la structure. Une classification hiérarchique est ensuite réalisée grâce à une de ces mesures, choisie par l'utilisateur.

Plus récemment, en 2018, afin de caractériser l'ensemble de Boltzmann d'ARN, Lin et al. [126] ont développé une nouvelle mesure de distance basée cette fois sur les appariements conflictuels et non les appariements communs.

Des méthodes à noyau sont également développées pour classifier les structures secondaires. Un noyau permet de représenter des données dans un espace vectoriel plus grand, appelé espace des caractéristiques, grâce à une fonction non-linéaire (le noyau). En 2008, Liu [128] propose un noyau, appelé *fuzzy kernel*, pour calculer une mesure de similarité basée sur la distance euclidienne. Le noyau utilise des caractéristiques de l'ARN qui sont extraites grâce à une décomposition en valeurs singulières des matrices à 6 dimensions représentant les ARN.

Un noyau, appelé NSPDK, a été développé par Costa et De Grave en 2011 [44] dans le but de classer les molécules chimiques. Pour cela, le noyau prend en entrée les graphes des molécules et calcule entre chaque paire de graphes une métrique basée sur la comparaison des paires de sous-graphes voisins. Ce noyau a été utilisé à plusieurs reprises pour la classification des structures secondaires d'ARN car il permet de comparer les motifs des structures et non simplement les appariements. Le noyau a tout d'abord été utilisé dans un outil appelé GraphClust [91] en 2012, puis dans un outil appelé RNAscClust [147] en 2017. Ces deux outils sont destinés à la classification des *reads* issus du séquençage.

Des outils, basés sur des méthodes de prédictions différentes, utilisent leur particularités pour la classification des structures qu'ils prédisent. Par exemple, un outil appelé RNAshapes [101], représente les structures en fonction de leur

topologie et de leur motifs en abstrayant la longueur et la l'emplacement des hélices. Les structures ayant la même topologie peuvent être classées ensemble. Un autre outil, appelé RNAHelices [94], prédit les structures d'ARN à partir d'un ensemble d'hélices et classe les structures grâce à une distance basée sur les hélices. Enfin, un outil basant également sa prédiction sur des hélices, appelé RNAprofiling [185], classe les structures prédites suivant la fréquence des hélices dans les structures.

Concernant la prédiction de structures de complexes d'ARN, une seule méthode a été développée. En 2016, Mneimneh et Ahmed [150] ont développé une méthode basée sur la classification ascendante hiérarchique et une mesure de distance calculant les différences entre des fenêtres de deux structures.

4.2 Notre méthode : C-RCPred

Comme nous l'avons vu dans l'état de l'art, aucune méthode prédisant des complexes d'ARN ne permet d'intégrer des contraintes utilisateurs ou des données structurales telles que le SHAPE. Nous présentons ici une nouvelle méthode pour prédire des structures secondaires de complexes d'ARN avec pseudonœuds de manière interactive, en intégrant des contraintes utilisateurs et des données structurales. Notre nouvelle méthode est basée sur celle du chapitre précédent, c'est-à-dire que nous cherchons à résoudre un problème de clique maximum, mais cette fois ci multi-objectif. Nous présentons donc dans un premier temps le problème de graphe multi-objectif permettant de trouver des combinaisons de structures et d'interactions correspondant à des complexes d'ARN. Nous y abordons la compatibilité entre les structures et interactions, le calcul des différents critères et comment réguler la profondeur des pseudonœuds. Nous présentons ensuite le programme linéaire multi-objectif associé et une heuristique multi-critère basée sur la recherche locale pour générer un ensemble de Pareto approché.

De plus, nous avons vu dans l'état de l'art qu'aucune méthode de classification de structures secondaires de complexes d'ARN n'existait. Nous proposons donc ici une nouvelle méthode basée sur un noyau et sur un algorithme de classification appelé classification spectrale.

4.2.1 Un problème de clique multi-objectif

Notre méthode est basée sur celle présentée au chapitre précédent, c'est-à-dire qu'à partir d'un ensembles de structures secondaires pour chaque ARN et d'un ensemble d'interactions pour chaque paire d'ARN, nous cherchons à trouver la meilleure combinaison formant un complexe. Dans notre nouvelle méthode, nous souhaitons intégrer des contraintes utilisateurs et des données structurales (SHAPE, DMS, PARS) grâce à l'optimisation multi-objectif. Pour cela, les meilleures complexes doivent correspondre à plusieurs critères :

— une énergie libre basse,
— le respect des contraintes utilisateurs,
— et l'accord avec les données structurales.

Les contraintes utilisateurs peuvent être l'ajout ou la suppression d'appariements ou de motifs (figure 4.2). Les motifs proposés sont les suivants : hélice, pseudonœud interne, pseudonœud externe, boucle terminale, boucle interne et boucle multiple. Pour chacun de ces motifs, l'utilisateur peut désigner les nucléotides concernés et/ou une longueur. Il peut également indiquer si un motif est présent, sans connaître de positions ou de longueurs. À chaque contrainte utilisateur est associé un indice de confiance variant entre 0 et 100 qui permet à l'utilisateur de pondérer la contrainte suivant le degré de connaissance qu'il a de cette contrainte.

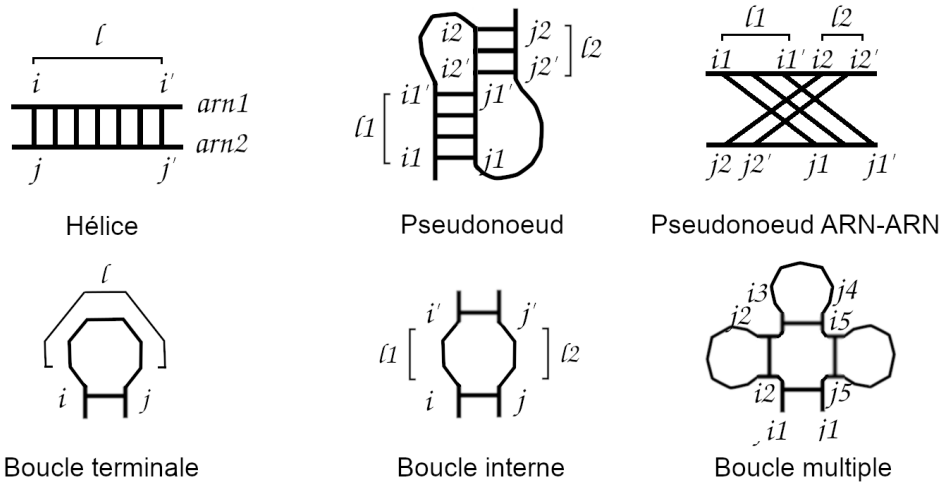

Figure 4.2 : Définitions des contraintes utilisateurs de motifs possibles pour un complexe d'ARN : hélice, pseudonœud interne, pseudonœud externe, boucle terminale, boucle interne et boucle multiple. Pour chaque motif sont indiquées les positions des nucléotides (i, j, ...) et les longueurs (l).

Comme au chapitre précédent, la prédiction de complexes est modélisée comme un problème d'optimisation combinatoire de graphe, plus particulièrement de recherche d'une clique. Cependant, il y a plusieurs différences dans le graphe, ce sont les suivantes :

— La compatibilité entre deux structures secondaires et/ou interactions : contrairement au chapitre précédent, un nombre de conflits est autorisé pour considérer que deux structures ou interactions sont compatibles.
— Deux poids supplémentaires (correspondant aux deux nouveaux critères) sont à maximiser sur les sommets : le respect des contraintes utilisateurs et l'accord des données structurales avec la structure/interaction correspondant au sommet.

Les cliques trouvées dans ce graphe correspondent à des complexes d'ARN. Contrairement au chapitre précédent, la profondeur des pseudonœuds potentiels de ces complexes d'ARN sera limitée à 3 : en effet, dans les bases de données, il n'a jamais été recensé de pseudonœuds avec une profondeur plus élevée.

Dans cette section, nous explicitons dans un premier temps la compatibilité entre deux structures ou interactions, comment sont calculés les différents

poids des sommets du graphe et comment limiter la profondeur des pseudo-nœuds.

Compatibilité entre les structures et les interactions

Deux structures et/ou interactions sont compatibles si la formation d'un complexe à partir d'elles n'engendre pas de conflits entre les appariements. En réalité, si quelques appariements sont en conflits, certaines bases peuvent se désapparier pour pouvoir former un complexe. Pour rendre compte de cela dans la modélisation de notre graphe, nous avons mis en place un seuil de compatibilité entre les structures et interactions. Ce seuil permet de moduler le nombre de conflits (nombre de nucléotides en conflits) en fonction du nombre de nucléotides impliqués dans les appariements des ARN communs aux deux structures/interactions, noté T (voir figure 4.3). L'utilisateur peut choisir un seuil allant de 80 à 100%. Si le seuil est inférieur à 80%, les structures/interactions sont incompatibles (score égal à 0). Le nombre de conflits est calculé de la manière suivante :

$$\text{Nombre maximum de conflits} = \left\lfloor \frac{\text{100-seuil}}{100} \times T \right\rfloor . \tag{4.3}$$

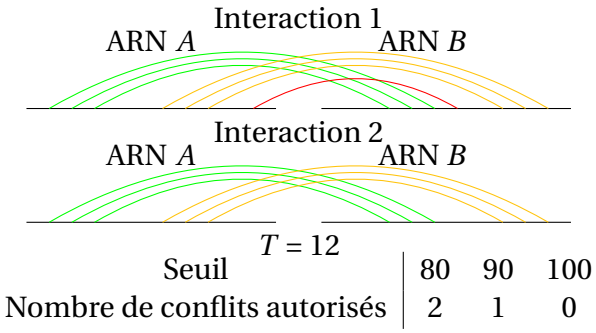

Figure 4.3 : Nombre de conflits autorisés en fonction de T et du seuil.

Dans les complexes prédits par notre méthode, nous souhaitons que chaque ARN ait au maximum une seule structure secondaire d'entrée, c'est-à-dire, que plusieurs structures secondaires d'entrée ne peuvent se "superposer" pour former la structure d'un ARN. Pour cela, nous fixons la compatibilité entre toutes les structures secondaires d'entrée correspondant à un même ARN à 0 : il n'y a donc pas d'arête entre elles. Toutes les autres structures secondaires d'entrée sont compatibles entre elles car elles ne s'influencent pas.

Lorsque des complexes sont générés avec des conflits, nous devons les réparer pour proposer à l'utilisateur des structures secondaires viables. Pour réparer une structure secondaire, nous créons toutes les structures secondaires possibles sans les conflits. Par exemple, si une structure secondaire possède un nucléotide engagé dans deux appariements, nous allons générer deux structures secondaires où chacune possède un des appariements. Ensuite toutes les structures secondaires générées sont classées suivant leur similarité, grâce à

la classification décrite en section 4.2.4, ce qui permet de ne retourner que la structure la plus représentative d'une classe.

Calcul des différents critères

Nous détaillons ici comment sont calculés les différents critères du problème multi-objectif, à savoir l'énergie libre, le respect des contraintes utilisateurs et l'accord avec les données structurales. Pour chaque structure secondaire et interaction d'entrée, nous calculons ces critères. Pour un complexe, la valeur du critère est égale à la somme des valeurs du critère des structures secondaires et des interactions composant le complexe.

L'énergie libre Le calcul de l'énergie libre pour chaque structure secondaire ou interaction d'entrée est réalisé soit par la librairie de ViennaRNA, soit par la librairie de NUPACK. Par défaut, c'est la librairie ViennaRNA qui est utilisée, avec le modèle de Turner [218] (avec les paramètres énergétiques de la mise à jour de 2004). Cependant, cette librairie ne prend pas en compte les pseudonœuds. Le modèle de NUPACK en revanche peut prendre en compte les pseudonœuds pour les ARN seuls (pas pour les interactions).

Les contraintes utilisateurs Le score des contraintes utilisateurs d'une structure ou interaction correspond à la somme des indices de confiance donnés par l'utilisateur pour chaque contrainte respectée. L'indice de confiance d'une structure se calcule de la manière suivante :

$$IC_{\text{structure}} = \sum_i IC_i, \tag{4.4}$$

avec IC_i l'indice de confiance d'une contrainte i respectée par la structure.

Les données structurales Nous considérons des données structurales pour chaque ARN de types SHAPE, PARS, DMS et leur versions haut-débits (voir section 1.1.6, page 1.1.6). Elles sont utilisées avec les structures secondaires uniquement. Les données structurales correspondent à la proportion de réaction des nucléotides à un produit chimique ou un traitement enzymatique. Cela donne une réactivité (normalisée en général entre 0 et 2.2, mais quelque fois entre 0 et 1) pour chaque nucléotide. Nous considérons dans la suite des données normalisées entre 0 en 1. Lorsque la réactivité tend vers 1, cela signifie que le nucléotide est probablement non-apparié. En revanche, lorsque la réactivité tend vers 0, cela signifie que le nucléotide est probablement apparié.

Une donnée de réactivité est associée à un nucléotide, or, notre problème est modélisé à l'échelle de structures. La définition d'un score indiquant la proportion du respect de ces probabilités pour une structure doit donc être défini. Si aucune donnée n'est connue, nous choisissons comme valeur par défaut 0.5, car une réactivité de 0.5 ne permet pas une interprétation.

Nous utilisons comme score le coefficient de corrélation de Matthews (MCC), introduit dans l'article [24] :

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}, \tag{4.5}$$

où TP est le nombre de vrais positifs (les nucléotides ayant une réactivité normalisée en deçà d'un seuil donné, fixé à 0.5, et impliqués dans un appariement Watson-Crick ou G-U), TN est le nombre de vrais négatifs (les nucléotides ayant une réactivité normalisée au dessus du seuil et non impliqués dans un appariement Watson-Crick ou G-U), FP est le nombre de faux positifs (les nucléotides ayant une réactivité normalisée en deçà du seuil et non impliqués dans un appariement Watson-Crick ou G-U), FN est le nombre de faux négatifs (les nucléotides ayant une réactivité normalisée au dessus du seuil et impliqués dans un appariement Watson-Crick ou G-U).

Avant le calcul du score, une étape de normalisation des données est nécessaire, puisque pour chaque ARN les expérimentations menées peuvent être différentes. Dans le cas du SHAPE, les données subissent une normalisation linéaire afin que les valeurs soient comprises entre 0 et 1 :

$$R(i) = \frac{r(i) - min(r)}{max(r) - min(r)}, \tag{4.6}$$

où $r(i)$ est la réactivité brute du nucléotide i et r le vecteur de toutes les réactivités de l'ARN.

Dans le cas de données DMS et PARS, il y a deux scores par nucléotide, correspondant à deux traitements des ARN. Dans le cas du DMS, il y a le contrôle et le traitement avec le produit DMS. Dans le cas de PARS, il y a le traitement avec l'enzyme V1 et le traitement avec l'enzyme S1. Pour déterminer la réactivité d'un nucléotide i, ces deux scores doivent être normalisés en fonction de l'abondance du transcrit et de sa longueur. L'abondance du transcrit correspond au nombre de *reads* obtenus lors de l'expérience, c'est-à-dire la somme de tous les scores. Une fois les deux scores normalisés selon l'équation 4.7, la différence des deux donne la réactivité finale (équations 4.8 et 4.9) [211].

$$R_{\text{exp}}(i) = \frac{ln(r(i)+1)}{\sum_{i=0}^{l} ln(r(i)+1)/l}, \tag{4.7}$$

où $r(i)$ est la réactivité brute du nucléotide i et l la longueur du transcrit.

$$R_{\text{DMS}}(i) = R_{\text{DMS}} - R_{\text{contrôle}}. \tag{4.8}$$

$$R_{\text{PARS}}(i) = R_{\text{S1}} - R_{\text{V1}}. \tag{4.9}$$

Limitation de la profondeur des pseudonœuds

Les complexes prédits par notre méthode RCPred peuvent avoir des pseudonœuds ayant une profondeur élevée (voir définition 1.1.1, page 13). Les structures d'ARN référencées dans les bases de données possèdent des pseudonœuds qui, pour la plupart, ont une profondeur égale à 2, voire à 3 dans de rares cas, nous souhaitons donc que la profondeur des pseudonœuds prédits par notre nouvel outil soit égale à 3 au maximum.

La profondeur des pseudonœuds générés par notre modélisation peut être importante si elle n'est pas limitée : cela est lié uniquement à l'assemblage d'interactions. En effet, nous autorisons au plus une structure secondaire d'entrée pour chaque ARN (voir section 4.2.1 sur la compatibilité), c'est-à-dire que

seuls les pseudonœuds initialement présents dans les structures secondaires d'entrée peuvent se retrouver dans les structures internes de chaque ARN des complexes. Par contre, comme plusieurs interactions peuvent s'assembler pour former un complexe, des pseudonœuds ayant une complexité importante peuvent être formés. La profondeur des pseudonœuds doit donc être contrôlée pour chaque interaction, c'est-à-dire pour chaque paire d'ARN possible.

Comme nous l'avons vu au chapitre 1, une interaction d'ARN peut être modélisée par un graphe de cercle (voir les définitions en section 1.2.2). Ce graphe va permettre de déterminer la profondeur des pseudonœuds de l'interaction, car cela revient à déterminer le nombre chromatique de ce graphe. Le nombre chromatique d'un graphe est le nombre minimum de couleurs utilisées pour la coloration d'un graphe dont les sommets voisins ne doivent pas avoir la même couleur. Déterminer le nombre chromatique d'un graphe de cercle est NP-difficile [193], nous allons donc utiliser une heuristique nous permettant d'obtenir de bonnes solutions. Il existe un algorithme glouton [35] de complexité en temps polynomial (voir algorithme 2) conçu pour déterminer une coloration optimale d'un graphe d'intervalles (le nombre de couleurs est égal au nombre chromatique). Comme les graphes d'intervalles sont similaires aux graphes de cercle, nous allons utiliser l'algorithme 2 comme heuristique pour déterminer une coloration.

Données : Un graphe d'intervalles ayant k sommets.
Résultat : Une coloration optimale du graphe.
1 On suppose que les k sommets sont ordonnés selon les extrémités gauches de leur intervalles, de façon croissante. Si deux intervalles sont identiques, ils sont ordonnés arbitrairement ;
2 Soient k couleurs $c_1, \ldots, c_k$;
3 **tant que** *le graphe n'est pas entièrement colorié* **faire**
4 Donner au prochain sommet non encore traité la plus petite couleur non déjà attribuée dans son voisinage ;
5 **fin**

Algorithme 2 : Coloration optimale d'un graphe d'intervalles [35].

4.2.2 Programme linéaire multi-objectif

Nous présentons ici le programme linéaire multi-objectif associé au problème de la clique maximum multi-objectif présenté précédemment. Nous définissons le graphe pondéré $G(V, E)$ formé par les structures secondaires et interactions d'entrée, tel que :

— V est l'ensemble des sommets représentant les structures secondaires et les interactions. Chaque sommet $v \in V$ a trois poids représentant l'énergie libre, le respect des contraintes utilisateurs et l'accord avec les données structurales.
— E est l'ensemble des arêtes représentant les compatibilités entre les sommets. Une arête existe si et seulement si deux sommets sont compatibles.

Chaque poids d'une clique est calculé par la sommes des poids des sommets composant la clique. Le détail du calcul de chaque poids est explicité en section précédente (voir section 4.2.1). Ici, nous cherchons la clique maximisant l'opposé de l'énergie libre et maximisant le respect des contraintes utilisateurs et l'accord avec les données structurales.

Nous modélisons ce problème sous la forme d'un PLNE. Notons x_v les variables de décisions binaires, valant 1 si le sommet $v \in V$ fait partie de la clique et 0 sinon. Soient $w_{E;v}$ l'opposé de l'énergie libre du sommet v, $w_{C;v}$ le poids correspondant au respect des contraintes utilisateurs et $w_{P;v}$ le poids correspondant à l'accord avec les données structurales.

Pour contrôler la profondeur des pseudonœuds, nous définissons pour chaque paire d'ARN $\{a,b\}$ un graphe $G_{\{a,b\}}(V_{\{a,b\}}, E_{\{a,b\}})$ où :

— $V_{\{a,b\}}$ est l'ensemble des hélices h_v contenues dans les sommets d'interactions $v \in V$ correspondant aux ARN a et b. Ces sommets ne sont pas pondérés.

— $E_{\{a,b\}}$ est l'ensemble des arêtes. Une arête existe entre deux sommets seulement si les deux hélices forment un pseudonœud (un croisement).

Pour chaque sous-graphe $G_{\{a,b\}}$ induit par les éléments de la clique (voir définition 1.2.4, page 23), le problème consiste à trouver une coloration pour laquelle le nombre chromatique est inférieur ou égal à une constante k, qui est le niveau maximum de profondeur pour les pseudonœuds. Nous fixons $k = 3$. Nous notons $y^{\{a,b\}}_{h_v;c}$ les variables de décision binaires du graphe $G_{\{a,b\}}$, égales à 1 si le sommet de l'hélice h_v (associé au sommet $v \in V$) est coloré avec la couleur c. La constante entière c est telle que $0 \leq c < k$. Le PLNE est le suivant :

$$\max f_1(x) = \sum_{v \in V} w_{E;v} x_v \tag{4.10}$$

$$\max f_2(x) = \sum_{v \in V} w_{C;v} x_v \tag{4.11}$$

$$\max f_3(x) = \sum_{v \in V} w_{P;v} x_v \tag{4.12}$$

tel que :

$$\sum_{u \in \overline{E_v}} x_u + |\overline{E_v}| x_v \leq |\overline{E_v}| \qquad \forall v \in V \tag{4.13}$$

$$\sum_{c} y^{\{a,b\}}_{h_v;c} = x_v \qquad \forall \{a,b\}, \forall h_v \in V_{\{a,b\}}, \forall v \in V \tag{4.14}$$

$$y^{\{a,b\}}_{h_v;c} + y^{\{a,b\}}_{i_u;c} \leq 1 \qquad \forall \{a,b\}, \forall (h_v, i_u) \in E_{\{a,b\}}, 0 \leq c < k \tag{4.15}$$

$$x_v, y^{\{a,b\}}_{h_v;c}, z^{\{a,b\}}_{c} \in \{0,1\} \qquad \forall v \in V, \forall \{a,b\} \tag{4.16}$$

où $\overline{E_v}$ est l'ensemble des sommets non-adjacents du sommet v, $\overline{E_v} = \{u | u \in V, (u,v) \notin E\}$ et $|\overline{E_v}|$ est le cardinal de $\overline{E_v}$.

Les fonctions objectifs, f_1, f_2 et f_3, visent, respectivement, à maximiser l'opposé de l'énergie, maximiser les contraintes utilisateurs et maximiser les données biologiques. Les contraintes 4.13 permettent de contrôler que les sommets désignés forment une clique. Les contraintes 4.14 assurent qu'à chaque hélice h_v du graphe $\{a,b\}$ est assignée au plus une couleur, si le sommet interaction $v \in V$ associé est dans la clique. Elles permettent de faire le lien entre le

problème de coloration et le problème de la clique. Les contraintes 4.15 forcent deux sommets adjacents à ne pas avoir la même couleur.

4.2.3 Heuristique multi-critère de recherche de clique

Nous présentons ici une heuristique pour résoudre notre programme linéaire multi-critère. En effet, notre programme linéaire consiste à résoudre le problème de la clique maximum multi-objectif qui est NP-difficile et très difficile à approximer (voir section 1.2.2, page 25). Le problème de la clique maximum nécessite donc des algorithmes ayant des complexités élevées pour le résoudre. Nous choisissons alors de le résoudre grâce à une heuristique.

Nous nous basons sur l'heuristique *Breakout Local Search* (BLS), décrite en détail en section 3.2.5, qui permet de résoudre le problème de clique maximum mono-objectif. Nous décrivons, ci-dessous, les modifications nécessaires pour rendre cette heuristique multi-critère (choix des mouvements dans le voisinage et génération du front de pareto). Nous expliquons également comment nous modifions l'heuristique pour pouvoir contrôler la profondeur des pseudonœuds. Enfin, nous présentons deux modifications pour améliorer l'heuristique : génération de la clique initiale et condition d'arrêt de la recherche.

Les modifications faites dans l'heuristique BLS sont les suivantes :

— **Choix des mouvements dans le voisinage :** dans l'heurisitique BLS, le mouvement apportant la meilleure amélioration à la solution courante est choisi grâce à la différence de poids apportée. Dans notre heuristique multi-critère, nous disposons d'un vecteur de poids, la meilleure amélioration correspond alors à choisir un mouvement dont la solution résultante domine l'ancienne ou n'est pas comparable avec elle.

— **Front de Pareto :** chaque nouvelle solution trouvée dominant les anciennes (ou qui lui sont non-comparables) est sauvegardée. Les solutions qui deviennent dominées par l'ajout de nouvelles solutions sont supprimées au fur et à mesure.

— **Contrôle de la profondeur des pseudonœuds :** lorsqu'un mouvement d'ajout d'un sommet, de remplacement d'un sommet, ou de réinitialisation de la clique, est candidat pour modifier la clique (la nouvelle solution domine ou n'est pas comparable avec celles du front de Pareto), la profondeur des pseudonœuds de la potentielle nouvelle solution est vérifiée. Pour cela, l'algorithme 2 est utilisé.

— **Génération de la clique initiale :** dans l'heuristique BLS, cette procédure consiste à sélectionner aléatoirement un sommet et ensuite à lui ajouter itérativement des sommets qui peuvent former une clique, jusqu'à ce qu'il n'y ait plus d'autres sommets qui puissent être ajoutés. Dans notre algorithme, cette procédure consiste à sélectionner aléatoirement un unique sommet constituant une clique. La clique grossira ensuite grâce à la phase de recherche locale.

— **Condition d'arrêt de l'heuristique :** dans l'heuristique BLS, la condition d'arrêt se produit soit lorsque la solution exacte est trouvée (si cette information est disponible), soit lorsqu'un nombre d'itérations maximum est atteint. Dans notre algorithme, nous avons choisi d'arrêter

la recherche lorsque nous constatons que celle-ci stagne, c'est-à-dire, lorsque la recherche est bloquée dans un optimum local et que malgré deux perturbations aléatoires, la recherche est toujours bloquée dans cet optimum.

4.2.4 Classification des structures secondaires des complexes

Comme nous avons pu le voir dans l'état de l'art, aucune méthode n'existe pour classer les structures secondaires de complexes d'ARN. Or, nos outils prédisent tous des ensembles de structures secondaires d'ARN qui peuvent être très similaires. Nous présentons donc ici une nouvelle méthode basée sur la classification spectrale à noyau. Nous présentons, dans un premier temps, le noyau servant à calculer la similarité entre deux complexes d'ARN, puis l'algorithme de classification spectrale.

Mesure de similarité

Pour obtenir une mesure de similarité entre nos structures secondaires qui ne soient pas basée simplement sur les appariements, nous avons choisi d'utiliser le noyau NSPDK [44]. En effet, ce noyau modélise les structures secondaires sous la forme de graphes permettant de l'utiliser directement pour des structures secondaires d'ARN. Les graphes sont construits de la manière suivante :

— un nucléotide correspond à un sommet ;
— une liaison (appariement ou squelette de l'ARN) est une arête.

Le noyau prend en entrée les graphes des molécules et calcule entre chaque paire de graphes une métrique basée sur la comparaison de paires de sous-graphes voisins. Cette métrique se calcule grâce à une relation, qui permet de sélectionner toutes les paires de sous-graphes voisins dans un rayon r de deux sommets u et v, dont les racines sont éloignées d'une distance d, dans un graphe G (voir figure 4.4). Grâce à cette relation, le noyau de décomposition va sélectionner ces paires pour deux graphes, pour un rayon r et une distance d donnés, et compter le nombre de paires identiques de graphes voisins. Enfin, le noyau NSPDK calcule la somme des noyaux de décomposition pour un ensemble de valeurs de r et de d paramétrées par l'utilisateur.

Algorithme de classification

Afin de réduire le nombre de structures secondaires similaires retournées par notre méthode, nous présentons ici une méthode de classification pour rassembler ces structures en classes et retourner uniquement un représentant par classe. De nombreux algorithmes de classification existent [244], nous pouvons citer, entre autres, les k-moyennes, la classification spectrale, les réseaux de neurones, les cartes auto-organisatrices (cartes de Kohonen), *etc.* Nos critères pour choisir un algorithme de classification étaient les suivants :

— l'algorithme devait être non-supervisé,
— il devait être possible de choisir le nombre de classes à l'avance.

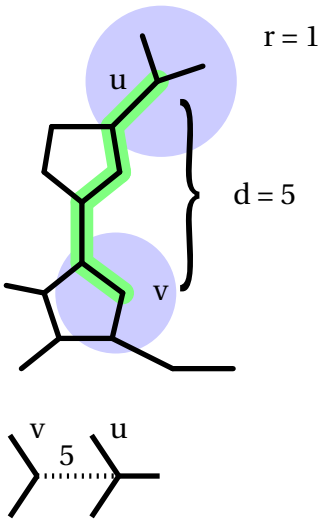

Figure 4.4 : Illustration de la relation $R_{r,d}$ du noyau développé par Costa et De Grave, pour $r = 1$ et $d = 5$. La relation permet de récupérer les sous-graphes issus des sommets u et v, éloignés d'une distance d, et pour un rayon r.

Notre choix s'est porté sur la classification spectrale qui est une méthode ayant de bonnes performances [154].

Le principe de la classification spectrale est de représenter les données (les structures d'ARN) sous la forme d'un graphe de similarités, ou d'une matrice de similarité, qui rassemblent les similarités entre chaque paire de données possibles (1-distance entre deux structures d'ARN). Á partir de ce graphe ou cette matrice on va chercher à trouver la coupe qui maximise les similarités entre les données de chaque classe et qui minimise les similarités entre les points de différentes classes. Pour cela, l'algorithme des k-moyennes (le principe de cet algorithme est expliqué ci-après) est effectué sur les k premiers vecteurs propres de la matrice de Laplace normalisée symétrique, notée L_{sym}, correspondant à ce graphe. Une matrice de Laplace d'un graphe décrit des informations sur la structure du graphe. Une matrice de Laplace, non normalisée, notée L, est calculée en faisant la différence de la matrice des degrés, notée D, et de la matrice d'adjacence du graphe, notée A :

$$L = D - A$$

La matrice des degrés d'un graphe est une matrice diagonale contenant les degrés de chaque sommet du graphe sur la diagonale et des 0 sinon. La matrice d'adjacence d'un graphe est une matrice carrée où chaque élément indique si deux sommets sont reliés ou non par une arête. Si deux sommets u et v sont reliés, l'élément de la matrice $A_{uv} = 1$, sinon $A_{uv} = 0$. La normalisation symétrique de la matrice de Laplace est réalisée de la manière suivante :

$$L_{sym} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Le but de l'algorithme des k-moyennes est de trouver k classes d'un ensemble de données (les structures d'ARN), représentées par une matrice de

données, en trouvant les k centres des classes. L'algorithme débute avec k centres aléatoire, et à chaque donnée est attribuée la classe dont le centre est le plus proche. L'algorithme est stoppé lorsque plus aucune modification ne peut être attribuée aux classes.

Les algorithmes de la classification spectrale [154] et des k-moyennes sont présentés ci-dessous (algorithmes 3 et 4 respectivement).

Données : Une matrice de similarités $S \in \mathscr{R}^{n \times n}$; le nombre de classes k.
Résultat : k classes.

1 Construire un graphe de similarités à partir de la matrice S ;
2 Calculer la matrice de Laplace normalisée symétrique L_{sym} de ce graphe ;
3 Calculer les k premiers vecteurs propres de L_{sym} ;
4 Soit $U \in \mathscr{R}^{n \times k}$ la matrice contenant les k premiers vecteurs propres en colonnes ;
5 Soit $T \in \mathscr{R}^{n \times k}$ la matrice telle que $t_{ij} = \frac{u_{ij}}{(\sum_k u_{ik}^2)^{1/2}}$: normalisation des lignes de U à la norme 1, telles que $||(t_{ij})_{1 \leq j \leq n}||_2 = 1$;
6 **pour** $i = 1, \ldots, n$ **faire**
7 | Définir $y_i \in \mathscr{R}^k$ étant le vecteur correspondant à la i^e ligne de T ;
8 **fin**
9 Classer les points $(y_i)_{i=1,\ldots,n}$ avec l'algorithme des k-moyennes en k classes (algorithme 4) ;

Algorithme 3 : Classification spectrale [154].

Données : Une matrice de données $S \in \mathscr{R}^{n \times m}$; le nombre de classes k.
Résultat : k classes.

1 Choisir aléatoirement k centres de classes $\{c_1, \ldots, c_k\}$;
2 **tant que** *les centres ne convergent pas* **faire**
3 | **pour** $i = 1, \ldots, k$ **faire**
4 | | Définir la classe C_i telle que C_i est l'ensemble des points de S qui sont les plus proches de c_i et non de c_j pour tout $j \neq i$;
5 | **fin**
6 | **pour** $i = 1, \ldots, k$ **faire**
7 | | Calculer c_i : moyenne des points de C_i ;
8 | **fin**
9 **fin**

Algorithme 4 : k-moyennes.

Le choix du nombre de classes est fait grâce à la méthode *eigengap*, décrite dans l'algorithme 5 [224]. Cette méthode consiste à représenter une matrice de données (les structures d'ARN) sous la forme d'un graphe de similarités et à calculer les valeurs propres de la matrice de Laplace normalisée symétrique associée à ce graphe. Les valeurs propres sont ordonnées de façon croissante. Le nombre de classes correspond au rang k pour lequel on observe le plus grand écart entre deux valeurs propres consécutives.

Notons que la structure secondaire représentant une classe est la structure la plus proche du centre de la classe. Il ne s'agit pas de la structure consensus de la classe. En effet, la structure consensus est une structure n'existant pas en

réalité, c'est pourquoi nous avons choisi la structure la plus proche du centre de la classe.

Données : Une matrice de données $S \in \mathscr{R}^{n \times m}$;
Résultat : Le nombre de classes k ;
1 Construire un graphe de similarités de la matrice S ;
2 Calculer la matrice de Laplace normalisée symétrique L_{sym} de ce graphe ;
3 Calculer les valeurs propres $\lambda_1, \lambda_2, \ldots$ de L_{sym} et les trier de manière croissante ;
4 Calculer $\arg\max(\delta_i)$ avec $\delta_i = |\lambda_i - \lambda_{i-1}|$, l'écart entre deux valeurs propres consécutives ;
5 $k = \arg\max(\delta_i)$;

Algorithme 5 : Méthode *eigengap* donnant un nombre de classes d'une matrice de données.

4.2.5 Outil C-RCPred

Cette section est dédiée à la présentation de notre outil implémentant notre méthode pour prédire des structures de complexes d'ARN en prenant en compte des données structurales et/ou des contraintes utilisateurs. Notre outil est appelé C-RCPred, pour *Constrained RNA Complex Prediction*. Nous avons implémenté C-RCPred en C++. Il prend en entrée des séquences d'ARN, un ensemble de structures secondaires pour chaque séquence d'ARN et un ensemble d'interactions ARN-ARN pour chaque paire de séquences d'ARN. Il peut également prendre en entrée des données SHAPE, PARS ou DMS, ainsi que des contraintes utilisateurs.

La première étape de l'outil est de calculer pour chaque structure et interaction d'entrée les différents critères selon la section 4.2.1 : l'énergie libre, le respect des contraintes et l'accord avec les données utilisateurs. Il utilise ensuite l'heuristique multi-critère, décrite précédemment, pour générer un ensemble de Pareto approché de cliques. Ces cliques sont transformées en complexes. Une clique peut correspondre à plusieurs complexes à cause de la compatibilité entre les structures secondaires/interactions qui est relâchée. Tous les complexes possibles sont générés, et donc, leur nombre peut être important. Ils sont alors classés, grâce à notre méthode basée sur la classification spectrale et le noyau NSPDK, et retournés à l'utilisateur.

L'outil est disponible sur la plateforme EvryRNA. Pour ce nouvel outil, nous avons mis en place un nouveau webserver, plus dynamique que celui de RC-Pred. En effet, le webserver doit être capable d'accepter des données structurales, ainsi que des contraintes utilisateurs. Le webserver de C-RCPred possède deux pages de formulaire et une page de résultats (voir figure 4.5).

La première page de formulaire consiste à télécharger les séquences d'ARN et à choisir les outils qui seront utilisés pour prédire les structures secondaires et les interactions d'entrée de C-RCPred. Les outils disponibles pour prédire les structures secondaires sont, pour le moment : BiokoP, pKiss, RNAsubopt. Ceux pour prédire les sites d'interactions sont : RNAduplex (ViennaRNA), RNAplex, RNAup (ViennaRNA), intaRNA, RIsearch. À terme, des outils acceptant des

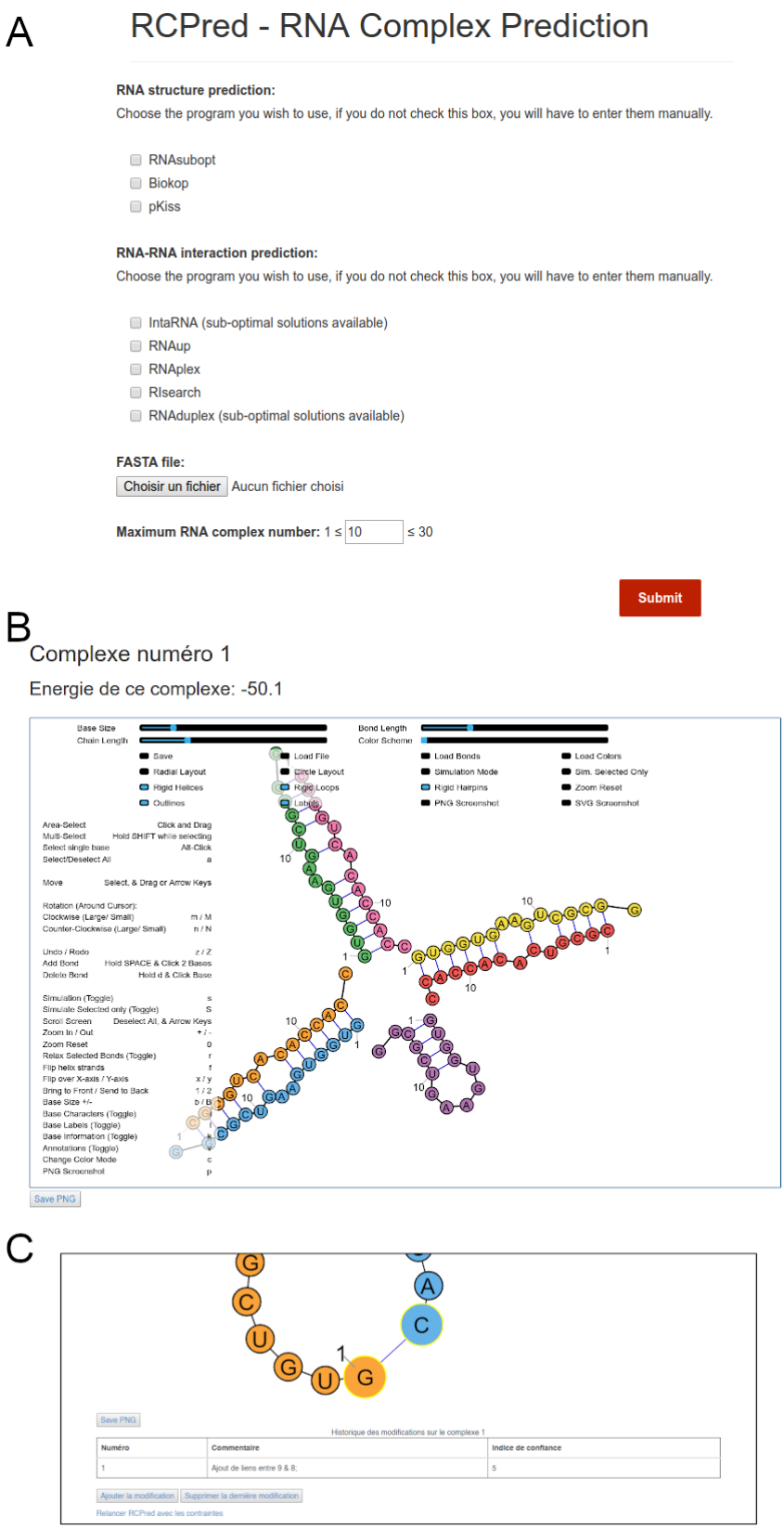

Figure 4.5 : Capture d'écran du webserver de C-RCPred. A) Page de formulaire n°1. B) Visualisation d'un complexe sur la page des résultats. C) Tableau d'ajout de contraintes sur la page de résultats.

données structurales telles que le SHAPE ou le PARS seront ajoutés (RME, ShapeKnots, Fold, RNApbFold, Rsample et RNAsc). L'utilisateur peut également choisir de n'utiliser aucun outil de prédiction s'il connait déjà des ensembles de structures qu'il voudrait utiliser pour la prédiction des complexes. C'est également sur cette page de formulaire que les données structurales et les contraintes utilisateurs sont ajoutées.

La seconde page de formulaire est également une page de résultats si des outils ont été choisis pour prédire les structures secondaires et/ou les interactions d'entrée de C-RCPred. On y trouve pour chaque ARN les structures prédites par les outils ainsi que la possibilité de modifier les structures, d'en ajouter ou d'en supprimer. Les mêmes fonctionnalités sont disponibles pour les interactions. Ainsi l'utilisateur est libre d'apporter ses propres structures et interactions, et/ou de modifier les résultats des outils de prédiction en fonction de ses connaissances sur les ARN qu'il étudie.

La page de résultats du webserver est composée, pour chaque complexe prédit par C-RCPred :

— d'un affichage dynamique de la structure, basé sur l'outil Ribosketch [133] ;
— d'un tableau de contraintes utilisateurs ;
— et d'une commande pour relancer C-RCPred.

Pour l'affichage dynamique de C-RCPred, nous avons utilisé l'outil Ribosketch, contrairement à RCPred, où nous avions utilisé l'outil forna [109]. Ribosketch est plus récent que forna et gère mieux l'affichage de structures de complexes d'ARN. L'affichage d'une structure de complexe grâce à Ribosketch permet d'afficher le complexe en deux dimensions suivant différents angles, de pouvoir colorer le complexe suivant les nucléotides ou les motifs et de pouvoir faire des captures d'écran. Ribosketch permet également d'interagir avec le complexe : déplacement, ajout ou suppression d'appariements, sélection de nucléotides, *etc.*

Lorsque les complexes sont affichés, l'utilisateur peut relancer C-RCPred avec des contraintes. L'utilisateur indique des contraintes via l'affichage dynamique des structures. Pour rendre cela possible, nous avons modifié l'outil Ribosketch pour enregistrer les contraintes de l'utilisateur via l'affichage. Dès qu'une contrainte est ajoutée sur la structure, elle est reportée dans le tableau. L'utilisateur peut ensuite supprimer une contrainte à partir du tableau. Ainsi, l'utilisateur peut suivre facilement les différentes modifications qu'il effectue dans la structure. Parmi les modifications de la structure affichée, l'ajout et la suppression d'appariements étaient déjà prévu par Ribosketch. En revanche, l'ajout et la suppression de motifs (voir figure 4.2) ne l'étaient pas, nous les avons donc ajoutés.

4.3 Résultats

Cette section est dédiée à la présentation des premiers résultats obtenus avec notre outil C-RCPred, nous permettant d'évaluer notre méthode. Nous présentons tout d'abord les jeux de données utilisés, puis les résultats obtenus pour chaque nouvelle fonctionnalité par rapport à l'outil RCPred. Dans un pre-

mier temps, nous nous intéressons à l'évaluation des nouvelles fonctionnalités : la compatibilité entre les structures et les interactions, au contrôle du niveau des pseudonœuds et la classification des résultats. Dans un second temps, nous nous intéressons aux nouvelles fonctions objectifs de notre programme linéaire : le respect des contraintes utilisateurs et l'accord avec les données structurales.

4.3.1 Jeu de données

Pour évaluer notre méthode, nous l'avons tout d'abord testée sur le jeu de données C_{multi} (présenté en section 3.3.1) afin d'estimer l'amélioration apportée par chaque nouvelle fonctionnalité.

Pour évaluer les nouveaux objectifs de notre méthode, le respect des contraintes utilisateurs et l'accord avec les données structurales, nous avons établi deux jeux de données qui seront décrits plus en détail dans la section 4.3.3.

4.3.2 Évaluation des nouvelles fonctionnalités

Cette section est dédiée à la présentation des premiers tests que nous avons effectués pour évaluer notre méthode. Nous avons évalué séparément les nouvelles fonctionnalités afin de déterminer l'apport de chacune. Nous nous intéressons tout d'abord à la compatibilité entre les structures et les interactions d'entrée pour former un complexe. Puis, nous présentons l'apport du contrôle de la profondeur des pseudonœuds. Enfin, nous testons notre méthode de classification.

Pour tester tous ces nouveaux paramètres sans que les uns n'interfèrent sur les autres, pour le test d'un paramètre, nous avons neutralisé tous les autres. Pour neutraliser la compatibilité, le seuil est fixé à 100, cela correspond à interdire tout conflit (voir figure 4.3).

Évaluation du paramètre de compatibilité

Nous avons testé l'apport du paramètre de compatibilité en exécutant C-RCPred avec un seuil de compatibilité variant entre 80 et 100, sur le jeu de données C_{multi}.

La figure 4.6 présente les MCC moyens et maximaux de 10 structures secondaires prédites par C-RCPred sur 10 exécutions sur le jeu de données C_{multi}. Le tableau 4.4 reporte le détail des sensitivités et PPV correspondants. Nous pouvons voir que, quelque soit le seuil, la qualité des structures secondaires prédites diminue quand le seuil de compatibilité diminue, aussi bien pour les statistiques moyennes que maximales. De plus, étant donné que même les statistiques maximales diminuent, cela veut dire que l'on n'augmente pas non plus la probabilité d'avoir une structure plus proche de la structure de référence parmi nos solutions en diminuant la compatibilité. Ce n'est pas exactement ce à quoi nous nous attendions. En effet, nous supposions que lorsque le seuil de compatibilité diminuerait, dans une certaine mesure, de nouvelles structures, plus proches de la structure de référence, pourraient apparaître, car parfois quelques conflits seulement peuvent empêcher des structures secondaires

d'ARN et des interactions de former un complexe. L'idée était donc d'autoriser ces quelques conflits. Bien sûr, si trop de conflits sont autorisés, des complexes très éloignés de la structure de référence sont générés, diminuant la qualité des résultats.

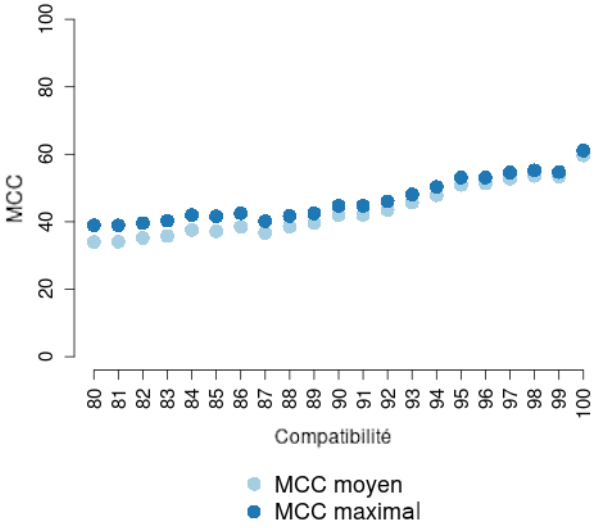

Figure 4.6 : MCC moyens et maximaux sur les 10e structures prédites par C-RCPred, en fonction du seuil de compatibilité sur le jeu de données C_{multi}. Les résultats sont calculés pour 10 exécutions de C-RCPred, puis la moyenne sur l'ensemble du jeu de données est effectuée pour les 10 premières structures retournées.

Compatibilité	80	81	82	83	84	85	86	87	88	89	90
Sensitivité moyenne	35.2	35.3	36.5	36.8	38.7	38.4	39.5	37.7	39.7	40.7	43.0
Sensitivité maximale	40.1	39.9	40.8	41.3	43.0	42.5	43.4	41.0	42.8	43.6	45.8
PPV moyen	34.7	34.8	35.8	36.6	38.3	37.9	39.3	37.4	39.2	40.2	42.6
PPV maximal	39.6	39.6	40.2	41.1	42.6	42.3	43.4	41.1	42.3	43.0	45.2

Compatibilité	91	92	93	94	95	96	97	98	99	100
Sensitivité moyenne	43.0	44.3	46.4	48.5	51.1	51.5	52.7	53.2	53.0	59.0
Sensitivité maximale	45.7	47.0	48.9	50.9	53.3	53.3	54.6	54.8	54.4	60.4
PPV moyen	42.7	44.3	46.4	48.8	52.3	52.7	54.2	55.4	55.1	61.6
PPV maximal	45.4	46.8	48.8	51.2	54.4	54.4	56.1	57.0	56.5	63.1

Tableau 4.4 : MCC, sensitivités et PVV, moyens et maximaux, des 10 premières structures de C-RCPred en fonction de la compatibilité sur le jeu de données C_{multi}. Les résultats sont calculés pour 10 exécutions de C-RCPred, puis la moyenne sur l'ensemble du jeu de données est effectuée.

Ici, nous voyons que même en autorisant quelques conflits (seuil à 99), la qualité globale des structures diminue. Cela pourrait venir du calcul de l'énergie. En effet, que les structures ou interactions soient compatibles à 100% ou non (jusqu'au seuil), le calcul de l'énergie libre d'un complexe est le même :

c'est la somme des énergies libres des structures et interactions le composant. Or, si les interactions et structures composant un complexe ne sont pas à 100% compatibles, c'est qu'il existe des conflits dans la structure et qu'elle doit donc être réparée pour supprimer ces conflits. Lors de la réparation, des modifications sont faites dans les structures et les interactions du complexe, donc l'énergie peut ne plus correspondre.

Nous pensons néanmoins que le paramètre de compatibilité est intéressant car cela permet de rendre plus réaliste la formation des complexes (quelques modifications peuvent se produire dans les structures lors de la formation d'un complexe) et que davantage de possibilités de structures de complexes sont potentiellement retournées par notre outil. Une solution pour améliorer les résultats pourrait être de mettre la valeur de l'énergie libre de deux sommets (interaction ou structure) sur l'arête les reliant. Cette valeur correspondrait à la différence d'énergie qu'il faut apporter lorsqu'un complexe est formé avec conflits par deux interactions (ou structures). Le poids d'une clique serait alors la somme des poids des sommets et des arêtes composant la clique.

Évaluation du contrôle du niveau des pseudonœuds

Pour évaluer l'apport du contrôle du niveau des pseudonœuds, nous évaluons l'outil C-RCPred et le comparons avec l'outil RCPred. Dans ces tests, nous n'utilisons ni contraintes utilisateurs, ni données structurales avec C-RCPred, ce qui nous ramène à un problème mono-objectif, comme dans RCPred. De plus, nous fixons le seuil de compatibilité à 100, ce qui revient à interdire tout conflit, comme dans RCPred. Les seules différences restantes entre les deux outils sont le contrôle du niveau des pseudonœuds et la génération de solutions de l'ensemble de Pareto approché retourné dans C-RCPred.

Nous avons évalué le niveau des pseudonœuds des structures prédites par C-RCPred et RCPred sur le complexe PDB_00851, dont la structure de référence est donnée en figure 4.7. Ce complexe est composé de 4 ARN auxquels nous nous référons par ARN 1, ARN 2, ARN 3 et ARN 4. Il possède deux pseudonœuds de profondeur 2.

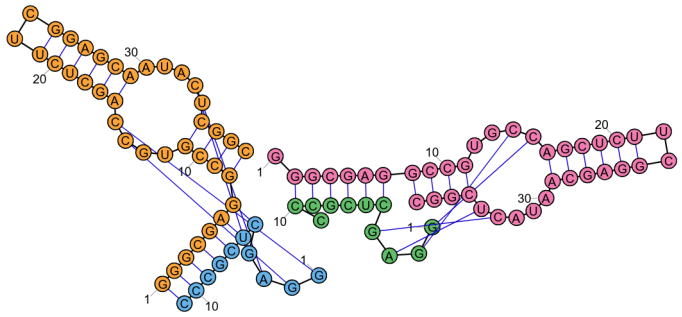

Figure 4.7 : Structure de référence du complexe PDB_00851. Bleu : ARN 1, orange : ARN 2, vert : ARN 3, rose : ARN 4. Figure réalisée grâce à Ribosketch [133].

Afin, d'étudier les profondeurs des pseudonœuds générés avec les outils

C-RCPred et RCPred, nous devons utiliser en entrée des structures secondaires et interactions qui peuvent s'assembler en formant des pseudonœuds assez élevés. Nous avons donc utilisé en entrée de C-RCPred et RCPred des structures secondaires et interactions qui permettent de prédire des structures de complexes avec un pseudonœud de niveau 4 (voir figure 4.8). Le but étant de voir si C-RCPred permet de générer des pseudonœuds avec une profondeur contrôlée (donc inférieure ou égale à 3), contrairement à RCPred qui peut générer des pseudonœuds ayant n'importe quelle profondeur.

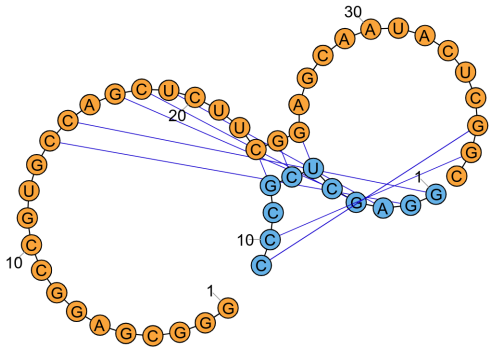

Figure 4.8 : Structure secondaire d'un exemple de pseudonœud de niveau 4 grâce à deux séquences d'ARN du complexe PDB_00851. Bleu : ARN 1, orange : ARN 2. Figure réalisée grâce à Ribosketch.

Nous avons exécuté les deux outils 10 fois avec ces entrées, puis nous avons étudié la profondeur des pseudonœuds présents dans les structures secondaires prédites. En moyenne, la profondeur des pseudonœuds des structures prédites par RCPred est de 4, tandis que celle de C-RCPred est de 3. Cela montre que la profondeur des pseudonœuds générés par RCPred peut être élevée et que le contrôle du niveau des pseudonœuds chez C-RCPred fonctionne correctement.

Évaluation de la classification des structures secondaires de complexe d'ARN

Nous étudions les résultats obtenus par C-RCPred avec la classification des structures secondaires de complexes d'ARN, afin de montrer que notre méthode de classification permet de réduire le nombre de structures similaires retournées par C-RCPred, tout en conservant la qualité des structures. Nous montrons dans un premier temps que les structures de complexes d'ARN sont correctement classées puis les résultats de la classification sur le jeu de données C_{multi}.

Qualité de la classification Pour tester si des structures de complexes d'ARN sont bien classées, nous avons testé cette méthode sur 30 matrices de similarités du noyau NSPDK dérivées de structures secondaires de 3 complexes d'ARN (le complexe carré, ainsi que les complexes B et C du splicéosome, voir figures 4.19 et 4.18). Les 30 matrices de similarités contiennent les similarités deux à deux d'un ensemble de structures de complexes d'ARN. Le nombre de

structures de complexes d'ARN pour chaque matrice varie entre 10 et 150. Pour déterminer si les structures sont correctement classées, nous avons représenté les matrices de similarités avec des cartes thermiques, nous permettant d'identifier le nombre de classes recherchées Puis, nous comparons avec le nombre de classes proposées par la méthode *eigengap*. La figure 4.9 montre la carte thermique d'une matrice du noyau NSPDK de 120 éléments correspondant à des complexes d'ARN (structures prédites à partir des séquences du complexe d'ARN carré, voir figure 1.16). Nous pouvons voir que deux classes se distinguent, et c'est également ce que nous propose la méthode *eigengap*. Cette méthode a permis de trouver un nombre de classes correct pour 26 matrices de similarités sur les 30.

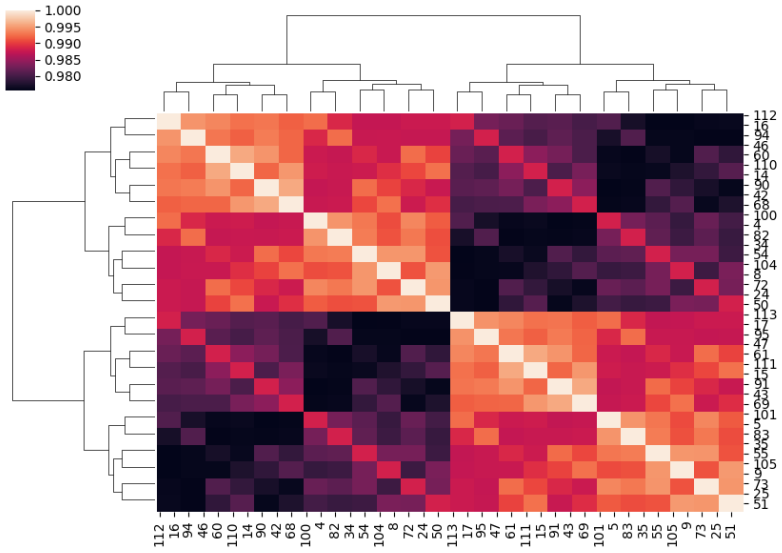

Figure 4.9 : Carte thermique d'une matrice du noyau NSPDK de taille 120 × 120. Carte réalisée à partir de 120 structures secondaires prédites du complexe carré.

Effet de la classification sur C_{multi} Nous montrons maintenant que notre méthode de classification permet de réduire le nombre de structures prédites par C-RCPred, tout en conservant leur qualité, sur le jeu de données C_{multi}. Pour cela, nous étudions les résultats obtenus par C-RCPred avec ou sans la classification.

La figure 4.10 montre le nombre de structures générées dans les deux conditions. Ce sont des moyennes sur 10 exécutions de l'outil. Comme nous pouvons le voir sur la figure, la classification réduit bien le nombre de structures générées. Pour la plupart des complexes, le nombre de structures retournées avec classification est égal à 1 en moyenne, c'est-à-dire qu'avant classification les structures générées sont toutes similaires pour ces complexes. En effet, dans cette exécution de C-RCPred sans contraintes utilisateurs et sans données structurales, le programme linéaire est mono-objectif, c'est-à-dire que les structures prédites sont choisies uniquement en fonction de leur énergie. Les

structures prédites ont donc toutes la même énergie et ont de fortes chances d'être similaires. Dans C-RCPred, le nombre de structures prédites augmente en modulant le seuil de compatibilité et en utilisant les autres objectifs.

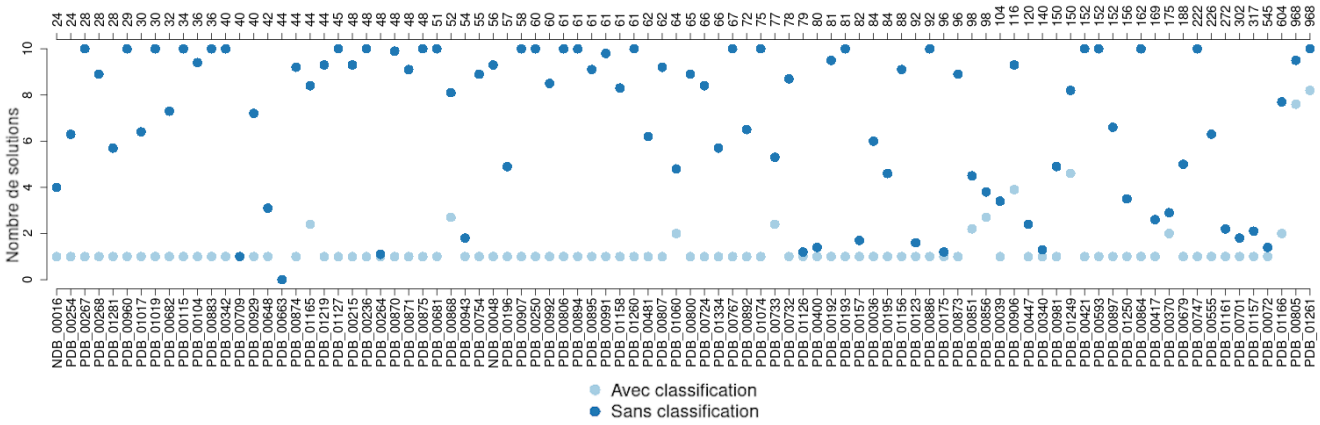

Figure 4.10 : Nombres moyens de structures secondaires obtenues par C-RCPred avec ou sans classification sur le jeu de données C_{multi}. Les moyennes ont été effectuées sur 10 exécutions pour chaque complexe. Nous avons récupéré au maximum 10 prédictions dans chaque condition.

La figure 4.11 présente les MCC moyens obtenus par les structures dans les deux conditions, pour 10 exécutions. Comme nous pouvons le voir, la qualité des structures prédites avec la classification ne diminue pas, la classification est donc efficace.

4.3.3 Évaluation des nouveaux objectifs

Nous présentons ici les premiers résultats que nous avons obtenus avec C-RCPred pour évaluer l'aspect multi-objectif. Nous présentons tout d'abord les résultats de nos premiers tests sur le nouvel objectif permettant de prendre en compte des contraintes utilisateurs, puis ceux sur le nouvel objectif permettant de prendre en compte des données structurales.

Ajout de contraintes utilisateurs

Pour évaluer l'apport des contraintes utilisateurs en tant qu'objectif supplémentaire, nous avons testé C-RCPred sur le complexe PDB_01165, dont la structure de référence est donnée en figure 4.12. Ce complexe possède 4 ARN auquel nous nous référons par ARN 1, ARN 2, ARN 3 et ARN 4. Les contraintes utilisées lors des tests sont décrites dans le tableau 4.5.

Trois de ces contraintes (a, b et c) sont des contraintes d'appariements : la contrainte a force un appariement entre les ARN 1 et 2, la contrainte b un appariement entre les ARN 3 et 4, et la contrainte c un appariement entre les ARN 2 et 3. Les contraintes a et b sont des appariements existant dans la structure de référence, elles doivent donc améliorer les résultats si elles sont respectées, au contraire de la contrainte c qui ne correspond pas à un appariement de la structure de référence et qui doit donc dégrader les résultats

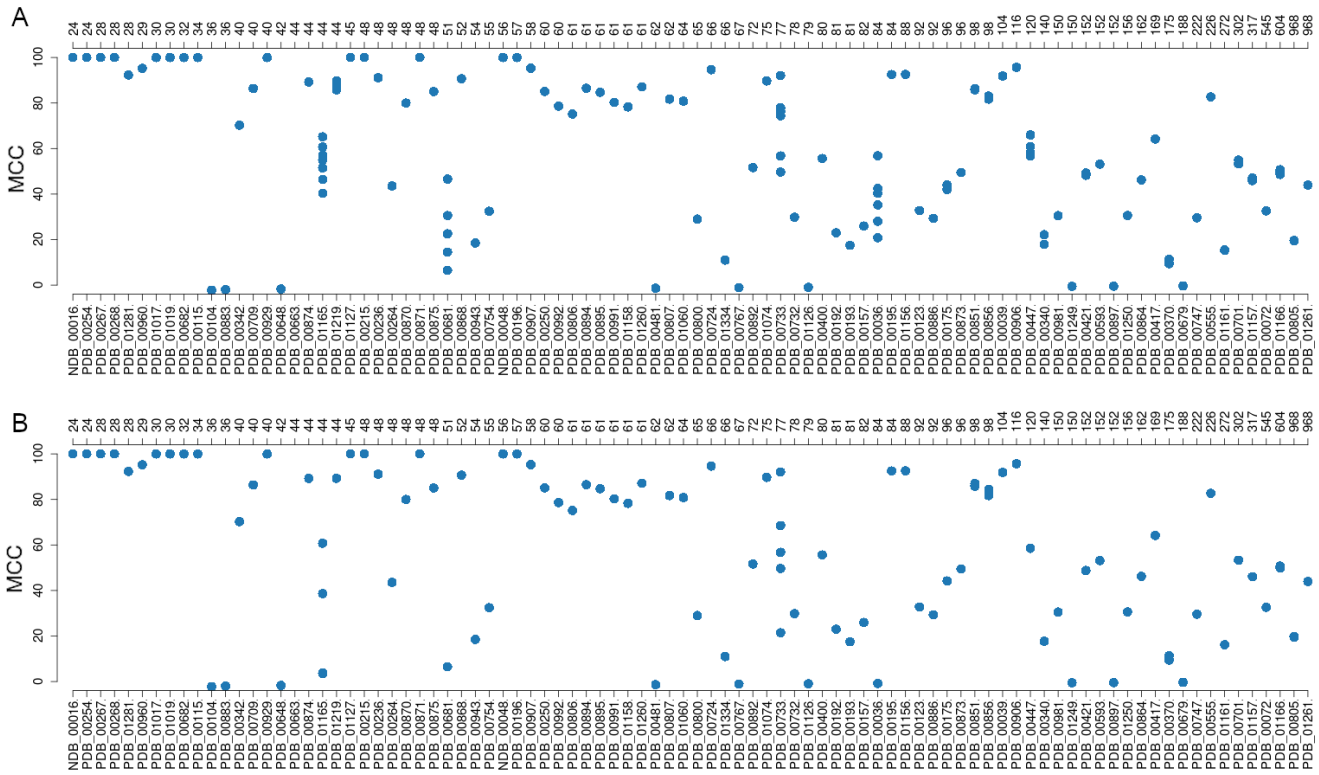

Figure 4.11 : MCC moyens de structures secondaires obtenues par C-RCPred sans classification (A) ou avec (B) classification sur le jeu de données C_{multi}. Les moyennes ont été effectuées sur 10 exécutions pour chaque complexe.

si elle est respectée. Les deux dernières contraintes (d et e) servent à tester d'autres types de contraintes, à savoir une contrainte de non-appariement (d) et une contrainte de boucle terminale (e). La contrainte d est une contrainte indiquant que le nucléotide en position 8 de l'ARN 2 ne doit pas être apparié. La contrainte e est une contrainte indiquant qu'une boucle terminale doit se trouver sur l'ARN 1. Ces deux contraintes ne correspondent pas à la structure de référence, donc si elles sont respectées, elles doivent dégrader les résultats.

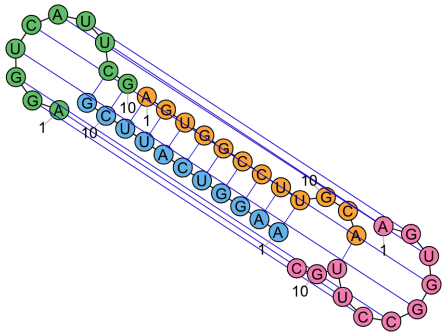

Figure 4.12 : Structure secondaire de référence du complexe PDB_01165. Bleu : ARN 1, orange : ARN 2, vert : ARN 3, rose : ARN 4. Figure réalisée grâce à l'outil Ribosketch.

Contrainte	Type	Positions
a	Appariement	ARN 1, base 9 avec ARN 2, base 1
b	Appariement	ARN 3, base 8 avec ARN 4, base 1
c	Appariement	ARN 2, base 8 avec ARN 3, base 1
d	Nucléotide non-apparié	ARN 2, base 8
e	Boucle terminale	ARN 1, base 4 à 9

Tableau 4.5 : Contraintes utilisateurs utilisées avec C-RCPred sur le complexe PDB_01165.

Nous avons exécuté C-RCPred soit avec une seule contrainte, soit avec plusieurs. Nous avons modulé l'indice de confiance entre 50 et 100. Un indice de confiance élevé indique que l'utilisateur est très confiant dans cette contrainte et qu'elle a de fortes chances d'être vraie. Les contraintes avec un indice de confiance élevé sont donc favorisées dans C-RCPred.

Les résultats des tests avec les contraintes *a* et *b* sont présentés en figure 4.13. Nous avons également reporté, comme contrôle, les résultats de C-RCPred sans l'utilisation de contraintes. Comme nous pouvons le voir, l'ajout des contraintes *a* et *b* permet d'augmenter la qualité des résultats. Les prédictions de C-RCPred avec les contraintes sont très proches de la structure de référence, avec un MCC moyen de 98% (utilisation des contraintes *a* et *b* ensemble), 95% (contrainte a seule) et 95% (contrainte *b* seule), pour tout indice de confiance utilisé. Même avec un faible indice de confiance, C-RCPred est bien orienté vers les prédictions les plus pertinentes. Cela est dû au changement de la nature du programme linéaire : dans la version dans contrainte, le programme est mono-objectif, tandis qu'avec les contraintes, le programme est bi-objectif.

Les résultats des tests avec la contrainte *c* sont présentés en figure 4.14 A. Nous pouvons voir que cette contrainte, ne correspondant pas à un appariement de la structure de référence, a bien pour effet d'éloigner les prédictions de la structure de référence, avec un MCC moyen proche de 0.

La figure 4.14 B montre les résultats de tests effectués avec les contraintes *a* et *c* ensemble. Ces deux contraintes partagent le même indice de confiance de confiance que nous faisons varier de 50 à 100. Quelque soit l'indice de confiance, nous pouvons voir que les prédictions proposées par C-RCPred ont un spectre large, ce qui correspond à l'alliance des deux contraintes, qui doivent améliorer (contrainte *a*) et dégrader (contrainte *c*) les résultats.

La figure 4.14 C présente les résultats de tests effectués avec les trois contraintes *a*, *b* et *c*, ensemble. Ces trois contraintes partagent le même indice de confiance que nous faisons varier de 50 à 100. Les contraintes correspondant à des appariements de la structure de référence (*a* et *b*), sont plus nombreuses que les contraintes ne correspondant pas à la structure de référence (*c*). Les contraintes *a* et *b* prennent donc le dessus sur la contrainte *c* et donc les prédictions de C-RCPred sont très proches de la structure de référence.

La figure 4.15 présente les résultats de tests effectués avec les trois contraintes *a*, *b* et *c*, ensemble, mais ayant cette fois-ci différents indices de confiance. Nous voulons étudier ici l'impact de l'indice de confiance sur un

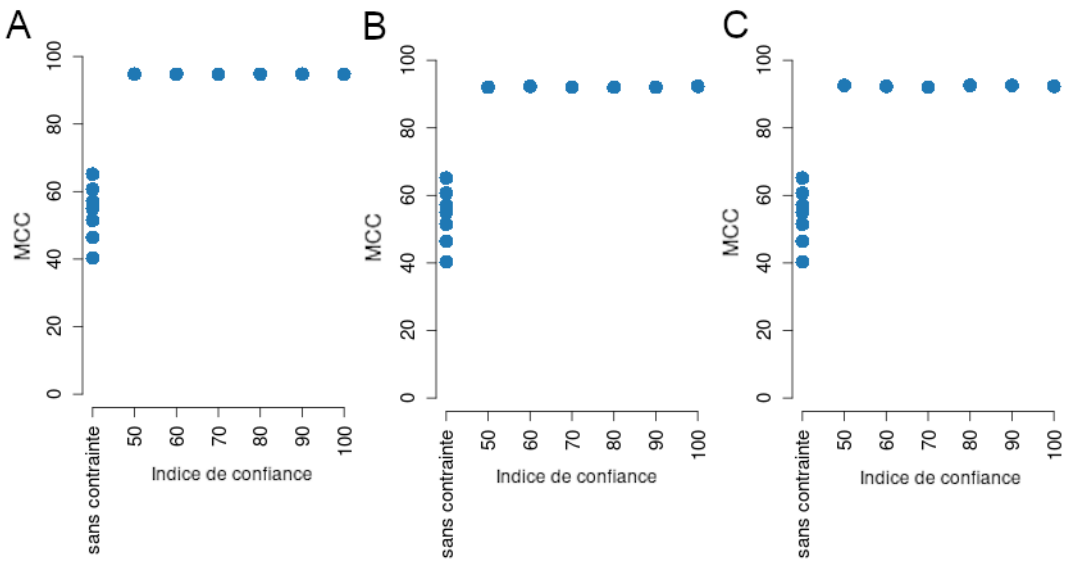

Figure 4.13 : MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs *a* et *b* sur le complexe PDB_01165. Le détails des contraintes est donné dans le tableau 4.5. Les résultats sont calculés pour 10 exécutions de C-RCPred, puis la moyenne est effectuée pour chacune des 10 premières solutions retournées. L'indice de confiance varie de 50 à 100. A) Les contraintes utilisateurs *a* et *b* sont utilisées et elles partagent le même indice de confiance. B) La contrainte *a* est utilisée seule. C) La contrainte *b* est utilisée seule.
La condition "sans contrainte" est un test contrôle.

ensemble de contraintes. Nous avons testé dans un premier temps les indices de confiance suivants : 90, 90 et 50 respectivement pour les contraintes *a*, *b* et *c*, notés (90;90;50). Nous n'avons fixé aucun indice de confiance à 100, pour ne pas utiliser les contraintes fortes, qui contraindraient trop les prédictions de C-RCPred. Les indices de confiance (90;90;50) appuient les contraintes *a* et *b* qui correspondent à la structure de référence et défavorise la contrainte *c* : cela permet à C-RCPred de prédire des structures très proches de la structure de référence.

Dans un deuxième temps, nous avons testé les indices de confiance suivants : 50, 50 et 90, respectivement pour les contraintes *a*, *b* et *c*. Ainsi, nous avons deux contraintes plus faibles totalisant un score de 100, correspondant à la structure de référence (*a* et *b*), contre une contrainte plus forte avec un indice de confiance de 90, ne correspondant pas à la structure de référence (*c*). Le panel des solutions est aussi bien réparti que dans les tests précédents dont les résultats sont présentés en figure 4.14 B, en effet, la force de chacune des contraintes (celles devant améliorer les prédictions, contre celle devant dégrader les prédictions) et leur nombre s'équilibrent.

Pour finir, nous avons considéré les contraintes *d* et *e* pour tester les autres types de contraintes. Ici, la contrainte *d* correspond à un non-appariement et la contrainte *e* à une boucle terminale. Ces deux contraintes doivent dégrader les résultats. Nous pouvons voir en figure 4.16 que c'est effectivement le cas. La

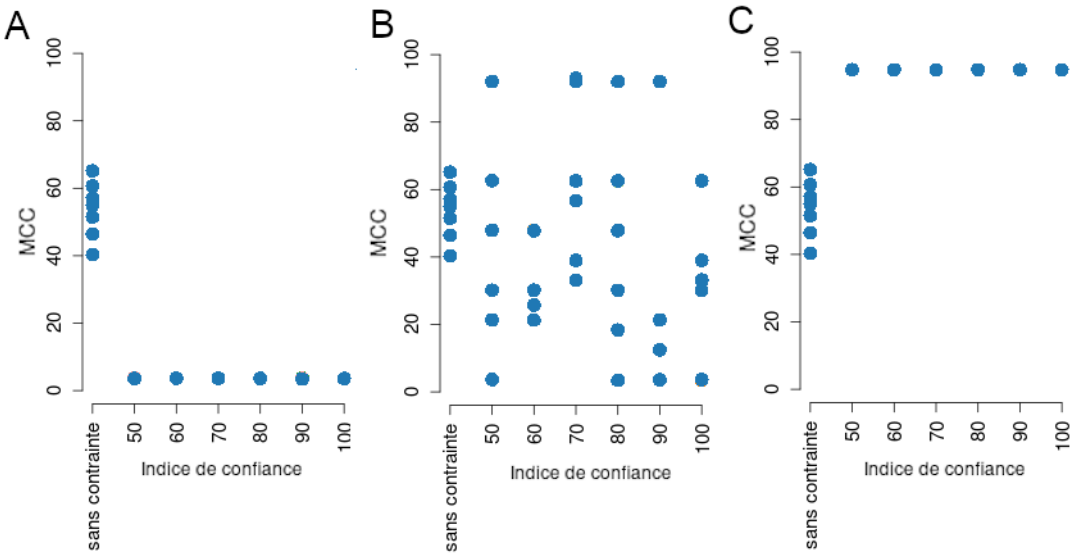

Figure 4.14 : MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs *a* et *b* et *c* sur le complexe PDB_01165. Le détails des contraintes est donné dans le tableau 4.5. Les résultats sont calculés pour 10 exécutions de C-RCPred, puis la moyenne est effectuée pour chacune des 10 premières solutions retournées. L'indice de confiance varie de 50 à 100. A) La contrainte utilisateur *c* est utilisée seule. B) Les contraintes *a* et *c* sont utilisées et elles partagent le même indice de confiance. C) Les contraintes *a*, *b* et *c* sont utilisées et elles partagent le même indice de confiance.
La condition "sans contrainte" est un test contrôle.

figure 4.17 montre des structures obtenues avec ces contraintes : comme nous pouvons le voir, ces structures respectent les contraintes *d* et *e*.

Modéliser le respect des contraintes utilisateurs en tant que fonction objectif permet de répondre correctement aux attentes des utilisateurs. Comme nous avons pu le voir lors des tests réalisés, lorsque des contraintes sont spécifiées par l'utilisateur, elles sont respectées par notre algorithme, en tenant compte de l'utilisation simultanée de plusieurs contraintes, ainsi que de leur différents indices de confiance.

Prise en compte de données structurales

Nous présentons ici les premiers résultats obtenus par C-RCPred pour la prédiction de structures des complexes pour lesquels nous avons trouvé des données structurales de PARS et de SHAPE. Nous présentons tous d'abord les données que nous avons pu récupérer, puis la qualité des structures et des interactions utilisées en entrée de C-RCPred et enfin, les résultats obtenus par C-RCPred.

Données utilisées Pour évaluer l'intégration des données structurales, nous avons rassemblé des données de structures secondaires de complexes validées expérimentalement et des données structurales de SHAPE et de PARS. Il a été

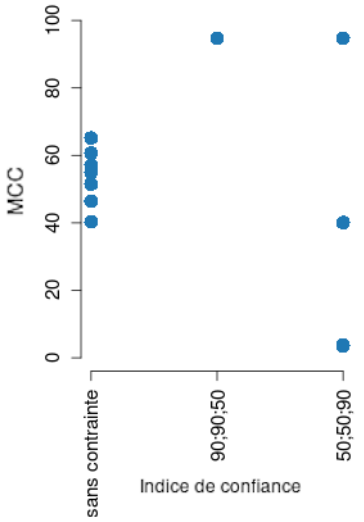

Figure 4.15 : MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs a et b et c, avec différents indices de confiance, sur le complexe PDB_01165. Les résultats sont calculés pour 10 exécutions de C-RCPred, puis la moyenne est effectuée pour chacune des 10 premières solutions retournées. Le détails des contraintes est donné dans le tableau 4.5. Les conditions 90;90;50 et 50;50;90 indiquent les indices de confiance des contraintes a;b;c pour les deux tests. La condition "sans contrainte" est un test contrôle.

très difficile de trouver des données de SHAPE ou PARS pour des ARN susceptibles de former des complexes de plus de deux ARN. Nous avons pu rassembler des données pour seulement 3 complexes que nous allons présenter.

Deux de ces complexes correspondent à deux structures que peut prendre le splicéosome qui est responsable de l'épissage chez les organismes eucaryotes (voir section 1.1.5, page 14). La structure 3D du splicéosome humain a été déterminée par cryo-microscopie électronique pour le complexe pré-catalytique, aussi appelé complexe B, par Bertram et al. en 2017 (voir figure 4.18). La structure du complexe actif, aussi appelé complexe C, a été déterminée par Zhang et al. en 2017 (voir figure 4.18), également par cryo-microscopie électronique. Les structures secondaires ont été déterminées à partir des structures 3D (figure 1.13). Les données structurales de PARS correspondantes ont été fournies par Wu et al. et effectuées à l'origine par Wan et al. [227].

La figure 4.18 permet de comparer les données de PARS à la structure de référence des complexes B et C. Les ARN du splicéosome changent leur structure fréquemment pour pouvoir effectuer l'épissage (figure 1.12), c'est en partie pour cela que les structures secondaires n'ont pas été prédites correctement avant la détermination expérimentale des structures 3D. Par conséquent, les données PARS peuvent ne pas être complètement en accord avec les structures de référence. La figure 4.18 montre que c'est effectivement le cas : il y a presque autant de données correspondant aux structures de références que de données ne correspondant pas. Nous allons tout de même tenter d'utiliser ces données afin de tester leur intégration dans notre outil.

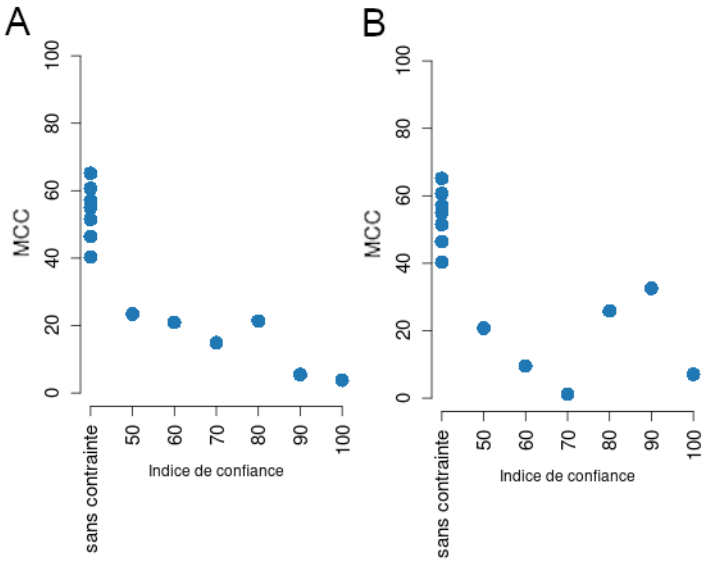

Figure 4.16 : MCC moyens des structures obtenues par C-RCPred avec les contraintes utilisateurs *d* et *e* sur le complexe PDB_01165. Les résultats sont calculés pour 10 exécutions de C-RCPred, puis la moyenne est effectuée pour chacune des 10 premières solutions retournées. Le détails des contraintes est donné dans le tableau 4.5. L'indice de confiance varie de 50 à 100. A) La contrainte utilisateur *d* est utilisée seule. B) La contrainte utilisateur *e* est utilisée seule.
La condition "sans contrainte" est un test contrôle.

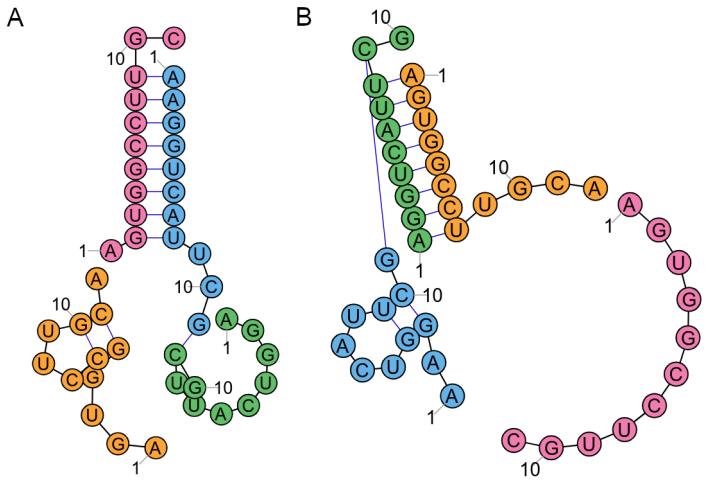

Figure 4.17 : Exemples de structures obtenues par C-RCPred avec les contraintes utilisateurs *d* (A) et *e* (B) sur le complexe PDB_01165. Bleu : ARN 1, orange : ARN 2, vert : ARN 3, rose : ARN 4. Figure réalisée grâce à l'outil Ribosketch.

Pour le complexe B, les données PARS correspondent, au mieux, avec l'ARN U5, en partie pour les ARN U4 et U2 et pas vraiment pour l'ARN U6. Pour le

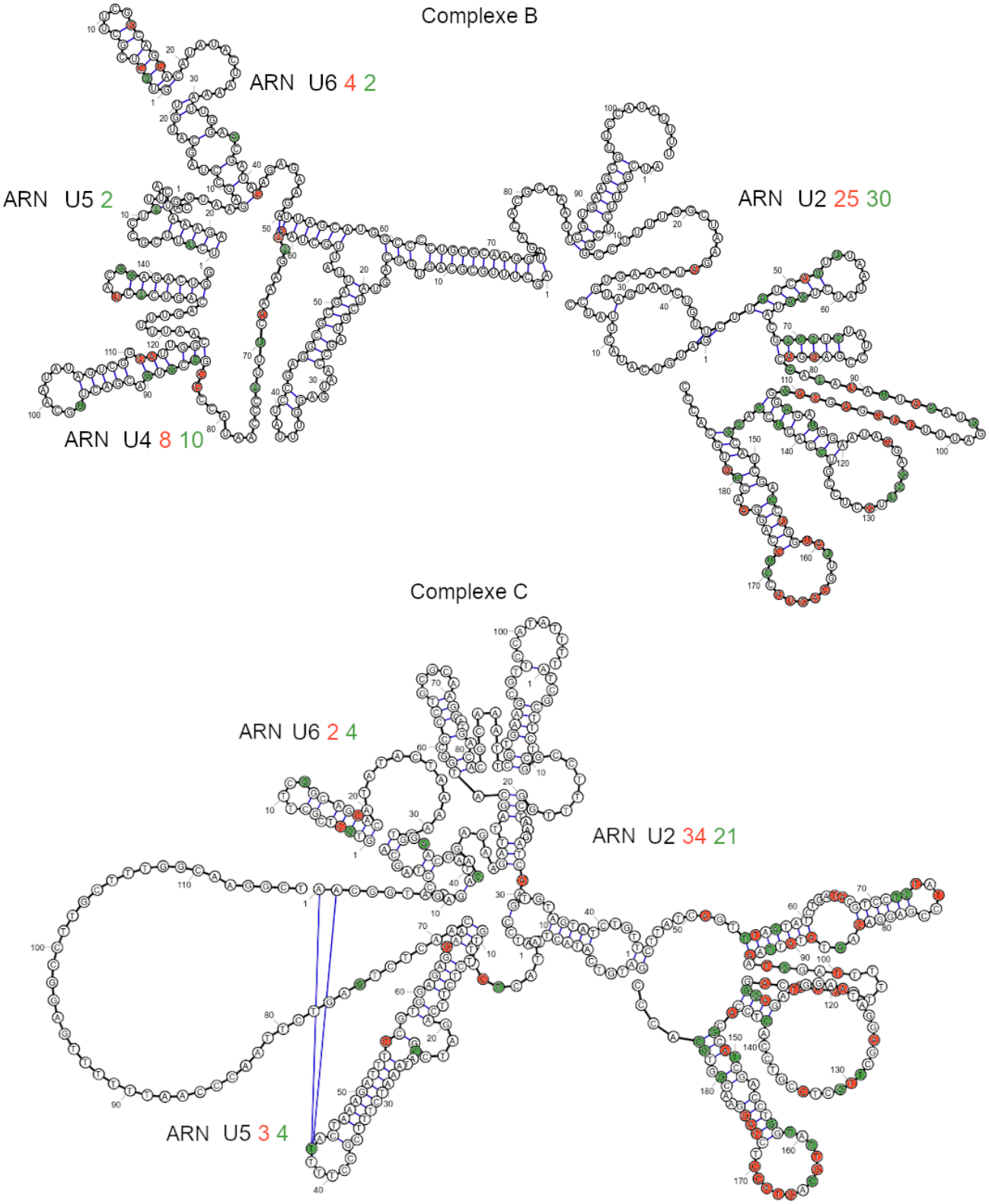

Figure 4.18 : Structures de référence des complexes B et C du splicéosome humain [20] colorées selon les données structurales de PARS [227]. Vert : la donnée est en accord avec la structure. Rouge : la donnée n'est pas en accord avec la structure. Image obtenue grâce à RiboSketch.

complexe C, les données PARS correspondent en partie pour les ARN U6 et U5 et pas vraiment pour l'ARN U2. Dans le but simplement de tester notre méthode, nous allons éviter d'utiliser les données pour l'ARN U6 du complexe B et pour l'ARN U2 du complexe C. Bien entendu, dans le cadre d'une prédiction de novo les données seraient utilisées telles quelles. L'idée est ici, de diminuer la possibilité d'avoir de mauvaises prédictions de structures à cause des données car alors on ne pourrait savoir si cela vient de notre méthode.

Nous avons également pu récupérer des données pour un complexe synthétique, appelé complexe "carré", dont les huit brins d'ARN s'auto-assemblent en carré [232] (figure 4.19). Bien que synthétique, ce complexe a été inspiré par un motif d'ARN qui se trouve chez le virus de l'hépatite C. Nous avons alors pu récupérer des données structurales de SHAPE dont les expériences avaient été effectuées sur le génome du virus de l'hépatite C [141]. Ces données proviennent de l'analyse du génome et non de l'analyse du complexe synthétique, elles peuvent donc différer de la structure de référence du complexe synthétique. La comparaison entre la structure de référence et les données structurales (figure 4.19) montre que les données SHAPE sont correctes pour le brin interne du motif formant le carré et qu'elles étaient à moitié correctes seulement pour le brin externe. Nous avons décidé, pour les mêmes raisons que précédemment, de considérer uniquement les données SHAPE du brin interne.

Bien entendu, des tests supplémentaires seront nécessaires pour tester notre intégration des données structurales en tant que fonction objectif.

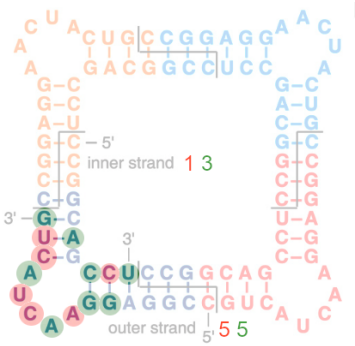

Figure 4.19 : Structure de référence du complexe d'ARN carré [57] colorée selon les données structurales de SHAPE [141]. Seuls 2 brins d'ARN sur les huit sont colorés car les autres brins sont identiques. Vert : la donnée est en accord avec la structure. Rouge : la donnée n'est pas en accord avec la structure.

Qualité des entrées Nous analysons ici les entrées que nous avons générées avec différents outils pour exécuter C-RCPred. Nous avons utilisé plusieurs outils pour générer ces entrées qui peuvent prendre en compte des données structurales ou non. L'idée est d'avoir des entrées différentes pour tester la prise en compte des données structurales dans C-RCPred. Ces conditions seront détaillées dans le paragraphe suivant, avec la figure 4.22.

Pour les structures secondaires, nous avons utilisé les outils suivants :

— si des données de SHAPE sont disponibles (ARN du complexe carré) : RME [240], ShapeKnots [86], Fold [183], RNApbFold [230], Rsample [204] et RNAsc [249],
— si des données de PARS sont disponibles (ARN des complexes B et C) : RME,
— sinon : RNAsubopt [130] et pKiss [101].

Pour les interactions, nous avons utilisé la version de l'outil intaRNA [148] qui peut prendre en compte des données SHAPE et l'outil RNAsubopt, ne prenant pas en compte de données structurales.

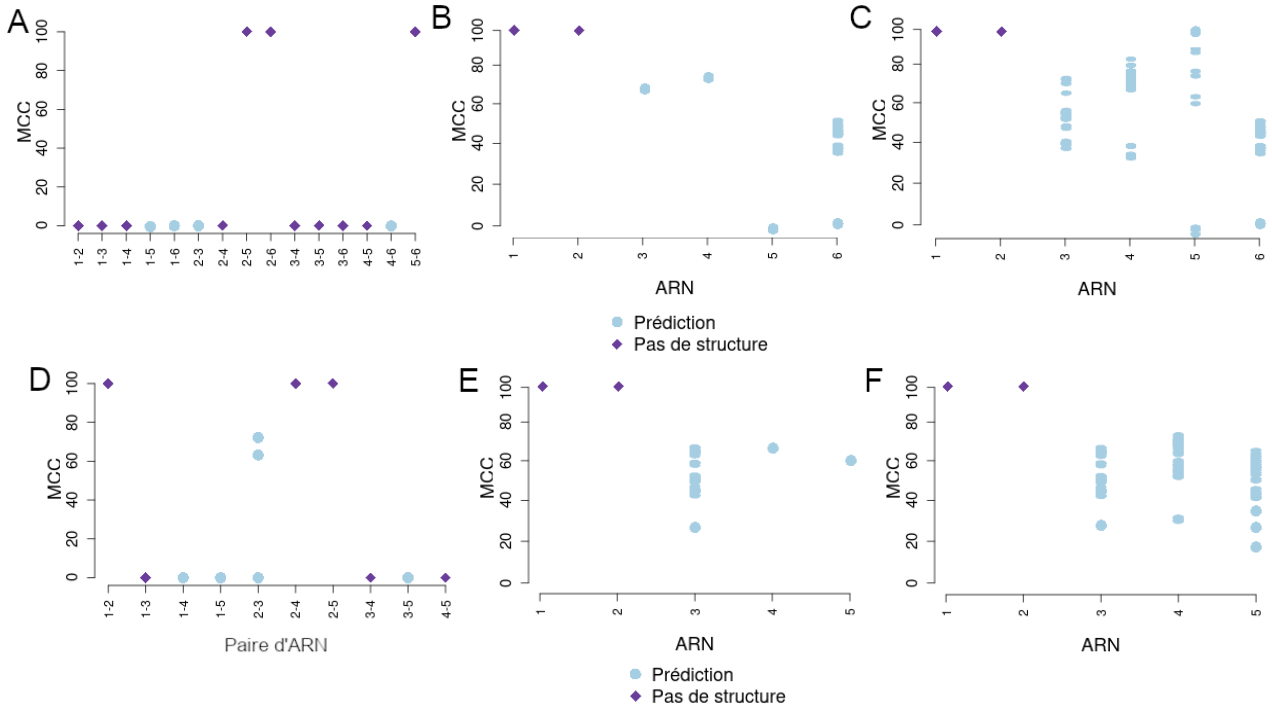

Figure 4.20 : MCC des interactions et des structures secondaires utilisées en entrée de C-RCPred pour les complexes B et C. A) Interactions prédites par l'outil RNAsubopt pour chaque paire d'ARN du complexe B. B) Structures secondaires prédites pour chaque ARN du complexe B par RME lorsque des données de PARS sont disponibles et par RNAsubopt et pKiss lorsqu'aucune donnée PARS est disponible. C) Structures secondaires prédites pour chaque ARN du complexe B par RNAsubopt et pKiss. D) Interactions prédites par l'outil RNAsubopt pour chaque paire d'ARN du complexe C. E) Structures secondaires prédites pour chaque ARN du complexe C par RME lorsque des données de PARS sont disponibles et par RNAsubopt et pKiss lorsqu'aucune donnée PARS est disponible. F) Structures secondaires prédites pour chaque ARN du complexe C par RNAsubopt et pKiss.
Lorsqu'un ARN ou une interaction n'a pas de structure et qu'une structure vide est renvoyée par l'outil de prédiction, nous indiquons un MCC égal à 100%, dans le cas contraire le MCC est égal à 0.

Dans le complexe B, très peu d'ARN interagissent : seulement 4 paires d'ARN interagissent sur 15. Les résultats pour ce complexe sont présentés en figure 4.20 A, B et C. Parmi les paires d'ARN interagissant, les ARN impli-

qués sont grands, ce qui diminue les chances pour que les sites d'interaction soient trouvés par l'outil de prédiction RNAsubopt (figure 4.20 A). Parmi les paires d'ARN n'interagissant pas, RNAsubopt prédit des sites d'interaction qui n'existent pas pour 8 paires d'ARN, ce qui va rendre plus difficile la sélection des interactions dans C-RCPred. Pour les structures secondaires générées, nous pouvons voir que les résultats diffèrent selon les ARN (figure 4.20 B et C). Grâce aux données structurales, nous pouvons voir que les meilleures structures sont sélectionnées pour les ARN 3 et 4 (figure 4.20 B). En revanche, ce n'est pas du tout le cas pour l'ARN 5, où la structure de référence est trouvée par l'outil de prédiction n'utilisant pas les données structurales mais pas par l'outil utilisant les données structurales. Cela peut s'expliquer par la qualité des données : en effet même si nous avons choisi d'utiliser seulement certaines données pour réduire le bruit, nous ne l'avons pas supprimé complètement. Pour l'ARN 6, aucune donnée n'était disponible, les résultats sont donc similaires. Quand les outils de prédiction sont utilisés sans données structurales, les prédictions sont beaucoup plus nombreuses et diversifiées.

Dans le complexe C, seulement 4 paires d'ARN sur 10 interagissent. Les résultats de ce complexe sont présentés en figure 4.20 D, E et F. Parmi les paires d'ARN interagissant, nous pouvons voir que nous obtenons des prédictions proches de la structure de référence seulement pour une paire d'ARN (figure 4.20 D). Pour les autres paires d'ARN, les MCC sont proches de 0. Pour les structures secondaires d'entrée, l'utilisation des données structurales permet de trouver les meilleures structures pour les ARN 4 et 5 (figure 4.20 E et F).

Les résultats du complexe carré sont présentés en figure 4.21. Pour ce complexe, nous avons pu tester un autre outil, appelé intaRNA pour générer des interactions, car il existe une version de cet outil prenant en compte des données SHAPE. Les résultats de l'outil intaRNA utilisé avec les données structurales sont présentés en figure 4.21 A et ceux de l'outil RNAsubopt sans données structurales en figure 4.21 B. Dans ce complexe, il y a 8 paires d'ARN qui interagissent et il y a deux types d'interaction, chaque type étant répété 4 fois. Nous pouvons voir que l'outil intaRNA n'a pas réussi à prédire de sites d'interaction avec une énergie favorable (figure 4.21 A). En revanche, l'outil RNAsubopt a pu déterminer parfaitement un des deux types d'interaction, ce qui correspond à 4 paires d'ARN sur les 8 qui interagissent (figure 4.21 B). En ce qui concerne les structures secondaires, dans ce complexe, les ARN sont petits et n'ont donc pas de structure interne. Les outils de prédiction retournent pour tous ces ARN aucune structure (figure 4.21 C et D).

Résultats de C-RCPred Pour évaluer la prise en compte des données structurales nous avons testé 4 conditions auxquelles nous nous référons par : DOUBLE SHAPE, SHAPE A, SHAPE B et SANS SHAPE ; et de manière similaire, nous notons DOUBLE PARS, PARS A, PARS B et SANS PARS, les 4 conditions avec les données PARS. Ces conditions permettent de savoir à quel niveau sont intégrées les données structurales, à savoir, soit dans C-RCPred, soit dans les outils utilisés pour générer les entrées de C-RCPred. Ces conditions sont les suivantes :

— Dans la condition DOUBLE SHAPE (ou DOUBLE PARS), nous utilisons

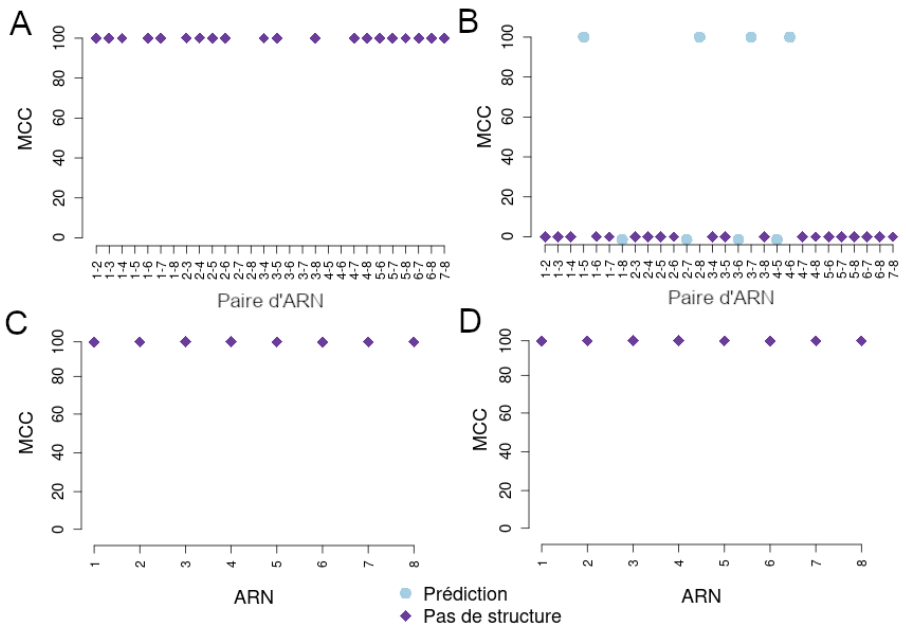

Figure 4.21 : MCC des interactions et des structures secondaires utilisées en entrée de C-RCPred pour le complexe carré. A) Interactions prédites par intaRNA pour chaque paire d'ARN. B) Interactions prédites par RNAsubopt pour chaque paire d'ARN. C) Structures secondaires prédites pour chaque ARN par les outils Fold, RNAfold, RNApbFold, RNAsc, Rsample, ShapeKnots, RestrainedMaxExpect lorsque des données de SHAPE sont disponibles et par RNAsubopt et pKiss lorsqu'aucune donnée SHAPE n'est disponible. D) Structures secondaires prédites pour chaque ARN par RNAsubopt et pKiss.
Lorsqu'un ARN ou une interaction n'a pas de structure et qu'une structure vide est renvoyée par l'outil de prédiction, nous indiquons un MCC égal à 100%, dans le cas contraire le MCC est égal à 0.

les données structurales avec C-RCPred et avec les outils utilisés pour générer les entrées.

— Dans la condition SHAPE A (ou PARS A), nous utilisons les données structurales uniquement avec les outils utilisés pour générer les entrées.
— Dans la condition SHAPE B (ou PARS B), nous utilisons les données structurales uniquement avec C-RCPred.
— Enfin, dans la condition SANS SHAPE (ou SANS PARS), qui est un contrôle, nous n'utilisons aucune donnée structurale que ce soit avec les outils utilisés pour générer les entrées, ou avec C-RCPred.

La comparaison entre les conditions SHAPE A, SHAPE B nous permet d'estimer si la prise en compte de données structurales par C-RCPred est meilleure que leur prise en compte par les outils servant à générer les entrées de C-RCPred. La comparaison entre les deux conditions SHAPE A, SHAPE B et la condition SANS SHAPE permet de déterminer si ces données structurales ont un impact sur les prédictions.

La liste des outils utilisés pour générer les entrées de C-RCPred en fonction des conditions est décrite en figure 4.22.

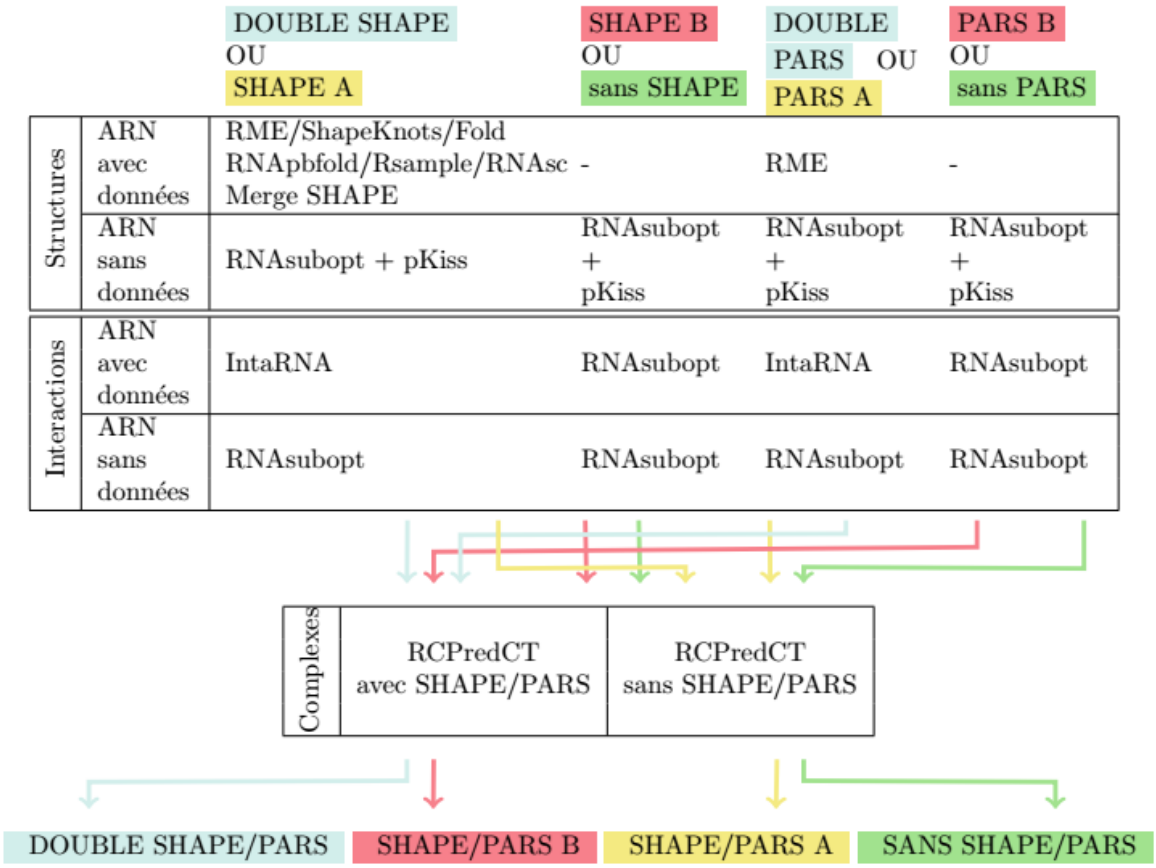

Figure 4.22 : Pipeline d'évaluation pour les complexes B, C et carré. Le tableau du haut indique la liste des outils utilisés pour générer les structures secondaires et les interactions utilisées en entrées de C-RCPred, suivant la disponibilité des données structurales. Le second tableau indique la prise en compte ou non de données structurales par C-RCPred. Les flèches de couleurs associent les outils utilisés pour générer les entrées avec C-RCPred et avec chaque condition : bleu : DOUBLE SHAPE/PARS, rouge : SHAPE/PARS B, jaune : SHAPE/PARS A et vert : SANS SHAPE/PARS.

La figure 4.23 reporte les résultats obtenus avec C-RCPred, NanoFolder, MultiRNAfold et NUPACK pour les complexes B, C et carré. Comme nous pouvons le voir, C-RCPred, tout comme les autres outils, ne réussit pas à prédire la structure de référence de ces complexes. Les MCC des structures prédites pour le complexe B varient entre 5 et 10% pour les conditions où C-RCPred a été utilisé avec des données structurales. La condition ayant les meilleurs résultats est SHAPE B, c'est-à-dire, la condition où C-RCPred est utilisé avec des données structurales et avec des entrées prédites avec des outils ne prenant pas en compte des données structurales. Ce qui correspond aux résultats précédent où la structure de l'ARN 5 avait été mal déterminée par les outils de prédiction prenant en compte les données structurales.

Les MCC obtenus pour les complexes C et carré sont proches de 0, ce qui ne rend pas les résultats interprétables pour l'intégration des données structurales. Ces résultats peuvent s'expliquer par une mauvaise qualité des structures et

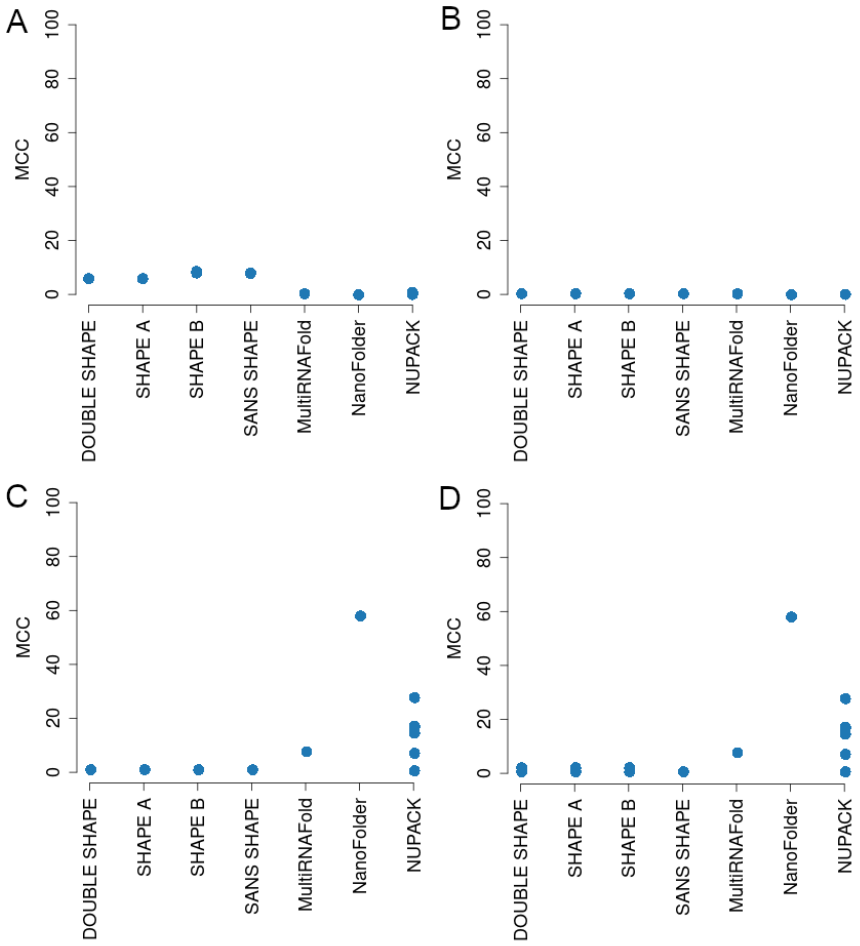

Figure 4.23 : MCC des structures prédites pour les complexes B, C et carré, par les outils C-RCPred, NanoFolder, MultiRNAfold et NUPACK. A : résultats du complexe B. B : résultats du complexe C. C : résultats du complexe carré avec intaRNA et RNAsubopt utilisés pour prédire les interactions d'entrée. D : résultats du complexe carré avec RNAsubopt utilisés pour prédire les interactions d'entrée.

interactions utilisés en entrée de C-RCPred.

Pour le complexe carré, en revanche les outils NanoFolder, MultiRNAfold et NUPACK obtiennent de meilleurs résultats que C-RCPred. Notamment NanoFolder qui détermine une structure avec un MCC proche de 60%.

Les structures secondaires des complexes B et C du splicéosome ont été déterminées avec les structures 3D grâce à la cristallographie. Leurs structures sont très dynamiques, ce qui les a rendu très difficile à prédire par les outils usuels de bio-informatique. Cela peut expliquer en partie les résultats obtenus. Nous ne pouvons pas conclure quant à la précision de l'intégration des données structurales dans l'outil C-RCPred.

4.4 Discussion

Nous avons développé un outil, appelé C-RCPred, basé sur un algorithme multi-objectif, pour la prédiction de structures secondaires de complexes d'ARN (composés de plus de deux ARN et pouvant inclure des pseudonœuds). À la différence de RCPred, décrit au chapitre précédent, C-RCPred est capable de prendre en compte des contraintes utilisateurs, ainsi que des données de biologie structurale telles que le SHAPE ou le PARS. Ces derniers sont assimilés à deux nouvelles fonctions objectifs supplémentaires, par rapport à RCPred qui cherche seulement à minimiser l'énergie libre. Les deux objectifs sont maximiser le respect avec les contraintes utilisateurs et l'accord avec les données de biologie structurale. L'utilisateur peut bien sûr choisir s'il ajoute ou non des contraintes et/ou des données structurales. Le programme linéaire associé devenant donc mono, bi ou tri-objectif selon les besoins.

Nous avons modifié l'heuristique utilisée dans RCPred afin de la rendre multi-objectif. La nouvelle heuristique permet de trouver une approximation (sans garantie de performance) de l'ensemble de Pareto tri-critère. Toutes les structures trouvées approchent les structures optimales. Pour le moment, nous ne générons pas d'ensembles de Pareto sous-optimaux approchés, car, étant donné que nous n'avons pas de garantie de performance, ils pourraient être très éloignés des ensembles de Pareto sous-optimaux exacts. Une perspective, dans un premier temps, serait donc d'estimer en moyenne l'écart entre la courbe de Pareto approchée générée par C-RCPred et la courbe de Pareto exacte. Puis, faire de même avec des ensembles de Pareto sous-optimaux.

Pour former un complexe, deux structures et/ou interactions doivent être compatibles. Dans l'outil RCPred, nous considérons que deux structures (ou interactions) sont compatibles s'il n'y a pas de conflits entre leurs appariements. En réalité, il est possible que quelques appariements se défassent pour qu'un ARN puisse interagir avec un autre. Dans C-RCPred, nous avons voulu modéliser ce phénomène en autorisant quelques conflits lors de la détermination de la compatibilité de deux structures ou interactions. Nous avons montré qu'en augmentant le nombre de conflits autorisés, la qualité globale des prédictions diminuait, et ce, même lorsque seulement quelques conflits sont autorisés. Nous pensons que notre modélisation induit un biais dans le calcul de l'énergie libre. Une perspective est de calculer la différence d'énergie libre lorsque deux structures/interactions ayant des conflits forment un complexe. Cette différence d'énergie serait indiquée sur l'arête reliant les deux sommets correspondants. L'énergie libre d'une clique serait alors la somme de l'énergie libre des sommets et des arêtes la composant.

Un défaut de l'outil RCPred est qu'il peut générer des pseudonœuds d'une profondeur importante, ne correspondant pas à des structures secondaires réalistes (profondeur supérieure à 3). Cela peut arriver lorsque plusieurs interactions se croisant sont associées pour former un pseudonœud. Ce n'est pas possible avec les structures secondaires car nous contraignons chaque ARN d'un complexe à avoir au plus une structure secondaire et que les outils que nous utilisons pour générer les structures secondaires d'entrée ne prédisent pas de pseudonœuds ayant des profondeurs supérieures à 3. Il s'agit donc tou-

jours de pseudonœuds externes. Dans C-RCPred, nous voulons donc contrôler le niveau de ces pseudonœuds. Ce problème peut être modélisé comme un problème de coloration de graphe de cercle, où le nombre minimum de couleurs utilisées est égal au niveau maximum de tous les pseudonœuds présents dans un complexe. Puis, nous avons utilisé comme heuristique un algorithme existant [35] permettant de trouver une coloration optimale d'un graphe d'intervalles (similaires aux graphes de cercle). Cela nous a permis de contrôler les solutions générées par C-RCPred : à chaque solution candidate à la courbe de Pareto, une vérification est faite pour déterminer la profondeur des pseudonœuds.

Dans C-RCPred, un des deux nouveaux objectif est l'accord avec les contraintes utilisateurs. Les contraintes utilisateurs permettent à l'utilisateur d'orienter les prédictions vers certaines structures. Nous avons implémenté deux sortes de contraintes : des contraintes fortes, qui doivent absolument être respectées et des contraintes faibles, qui orientent les prédictions sans les contraindre complètement. Nous avons montré que modéliser les contraintes faibles sous la forme d'une fonction objectif permet d'obtenir des prédictions respectant bien les contraintes suivant leur indice de confiance. Ces contraintes rendent également la prédiction plus interactive, notamment grâce à l'interface graphique dynamique qui permet de visualiser les complexes prédits en ajoutant des contraintes. L'utilisateur peut alors relancer les prédictions avec de nouvelles contraintes.

Le troisième objectif de notre problème de prédiction est l'accord avec des données structurales, telles que le SHAPE, le PARS ou le DMS. Trouver de telles données pour des complexes d'ARN fut très difficile ; en effet, la plupart des données sont disponibles pour un seul ARN. Nous avons pu trouver des données notamment pour certains ARN du splicéosome grâce à des études à échelle transcriptomique. Cependant, même avec ces données, tout le transcriptome n'est pas couvert. De plus, le splicéosome est un complexe dont la structure secondaire est difficile à prédire car c'est un complexe très dynamique. Par conséquent, les structures secondaires et interactions utilisées en entrée ne sont pas d'assez bonne qualité pour permettre une prédiction correcte du complexe. Cela nous a empêché de déterminer l'apport des données structurales en tant que fonction objectif supplémentaire. Une perspective est donc de tester sur de nouvelles données et de tester également avec des données artificielles pour évaluer notre intégration de données structurales.

Nous avons également abordé dans ce chapitre la classification des structures secondaires de complexes d'ARN. En effet, que ce soit avec RCPred ou C-RCPred, le nombre de structures retournées peut être important et surtout, ces structures peuvent être très similaires. De plus, suivant le seuil de compatibilité, le nombre de structures similaires peut être décuplé. À notre connaissance, il existe actuellement uniquement des outils permettant la classification de structures secondaires d'ARN et non de complexes d'ARN. Nous avons donc développé une méthode de classification de structures secondaires de complexes d'ARN basée sur une méthode classant les structures secondaires de complexes d'ARN avec un noyau appelé NSPDK. Ce noyau a été créé initialement pour calculer des distances entre des graphes de molécules chimiques. Nous avons

utilisé ce noyau avec la classification spectrale et nous avons montré que notre méthode permet de classer correctement les structures. Une perspective est de proposer un outil indépendant de C-RCPred proposant la classification d'un ensemble de structures secondaires de complexes d'ARN.

5 Résolution de programmes linéaires en nombres entiers bi-objectifs

Ce chapitre est dédié à la présentation de nouvelles méthodes de génération d'ensembles de Pareto. Plusieurs méthodes génériques ont été proposées afin de trouver les k meilleures solutions d'un problème d'optimisation mono-critère. De même, plusieurs méthodes génériques ont été proposées dans la littérature afin d'obtenir la frontière de Pareto, correspondant aux solutions optimales, d'un problème d'optimisation multi-objectif (voir section 1.2.3, page 27). Cependant, aucune de ces méthodes n'était adaptée à notre application en biologie, car nous avions besoin de trouver les k meilleurs ensembles de Pareto d'un problème d'optimisation bi-objectif, correspondant à des solutions sous-optimales. Nous présentons ici un nouvel algorithme nous permettant de générer les k meilleurs ensembles de Pareto exacts. Il est important de faire la distinction entre l'ensemble de Pareto et le front de Pareto, ce dernier contenant l'ensemble des valeurs atteintes par les fonctions objectifs pour les solutions de l'ensemble de Pareto, tandis que l'ensemble de Pareto contient les solutions. La plupart des méthodes existantes s'attachent à trouver le front de Pareto, mais certaines permettent de retrouver facilement l'ensemble de Pareto.

Notre algorithme est basé sur la méthode ϵ*-constraint* et effectue une recherche dichotomique dans l'espace des solutions. À chaque étape de la recherche, nous résolvons une version mono-objectif du problème bi-objectif pour trouver une solution. Ce problème mono-objectif vise à optimiser une des fonctions objectifs, tout en maintenant la seconde entre deux seuils. Nous déterminons ensuite à quel ensemble de Pareto appartient la solution trouvée, puis nous lançons deux nouvelles recherches en définissant deux nouveaux seuils pour chaque recherche : en dessous et au dessus de la solution. La recherche s'arrête une fois que toutes les solutions appartenant aux k meilleurs ensembles de Pareto recherchés ont été trouvées.

La complexité de résolution des programmes linéaires en nombres entiers est exponentielle. Nous proposons donc également une version parallélisée de notre algorithme. De plus, nous proposons également une version approchée de notre algorithme, où nous supposons que les programmes mono-objectifs à

résoudre peuvent l'être de façon approchée. Comme notre algorithme est en partie basée sur la méthode *ϵ-constraint*, nous présentons aussi une version approchée de cette méthode.

Enfin, nous nous sommes intéressés à un problème connexe. Étant donné un ensemble de solutions non triées, nous cherchons à trier ces solutions de manière à obtenir les k meilleurs ensembles de Pareto. Nous proposons dans ce chapitre un algorithme exact pour résoudre ce problème.

Nous avons ici axé notre recherche sur la programmation linéaire bi-objectif. En effet, nos algorithmes sont dédiés à trouver des solutions à des programmes linéaires bi-objectifs. Cependant, nous aimerions faire remarquer que nos algorithmes sont plus général : ils peuvent également s'appliquer à tout autre problème d'optimisation bi-objectif et non pas seulement aux programmes linéaires.

Ce chapitre est donc consacré à la présentation d'un nouvel algorithme générique déterminant les k meilleurs ensembles de Pareto d'un programme linéaire en nombres entiers (PLNE), à son analyse et à sa parallélisation. Ce chapitre traite également d'algorithmes approchés pour déterminer les k meilleurs ensembles de Pareto d'un PLNE.

5.1 État de l'art

Plusieurs approches de résolution d'un PLNE bi-critère existent [70]. On peut distinguer ainsi :

— Les méthodes dites Pareto qui cherchent à déterminer le front de Pareto, soit de manière exacte, soit de manière approchée.
— Les méthodes consistant à se ramener à un problème mono-objectif par une combinaison linéaire des fonctions objectif. Ces méthodes sont les plus simples mais elles ne permettent de ne trouver qu'une seule solution. Il est possible d'itérer la résolution en faisant varier la combinaison linéaire, mais dans ce cas, on n'obtient qu'un sous-ensemble du front de Pareto.
— Les méthodes consistant à chercher une solution qui soit la plus proche possible du point idéal (voir définition 1.2.19).
— Les méthodes consistant à traiter l'une des fonctions objectifs comme une contrainte.

Dans la suite, nous nous intéressons uniquement aux méthodes permettant de résoudre les problèmes en nombres entiers. Nous verrons dans un premier temps les méthodes exactes, puis les méthodes approchées et les méta-heuristiques. Nous discuterons également des méthodes permettant de trouver des solutions sous-optimales à des problèmes mono-critères.

5.1.1 Obtention de courbes de Pareto exactes

Les méthodes exactes nécessitent de résoudre plusieurs fois le problème entier pour obtenir un front de Pareto, ce qui est coûteux en temps. La plupart se ramènent à un problème mono-objectif, comme la méthode *ϵ-constraint* [21], décrite ci-dessous, qui permet de trouver le front de Pareto d'un problème

bi-critère. Cette méthode a été généralisée, notamment pour les problèmes avec, strictement, plus de deux objectifs [142, 143]. Des variations ont également été proposées pour améliorer le temps d'exécution de l'algorithme par des méthodes récursives [161, 162, 111, 252].

Une méthode, appelée méthode des deux phases [221], permet de résoudre, elle aussi, des problèmes bi-critères. La première phase consiste à générer l'ensemble des solutions supportées (voir définition 1.2.18) en utilisant le théorème de Geoffrion. La deuxième phase permet de générer l'ensemble des solutions non supportées dans le triangle supérieur de chaque paire de solutions non supportées (en supposant que c'est un problème où les deux critères doivent être minimisés). Cette méthode a été généralisée à un problème d'affectation (un problème d'affectation consiste à affecter des tâches à des agents) à trois critères [170].

Une méthode, appelée 2-*parallel partitioning method* (PPM) [124], permet de résoudre des problèmes bi-critères en trois étapes. La première étape permet de réduire l'espace de recherche en trouvant les solutions extrêmes du front de Pareto. La deuxième étape permet de trouver les solutions supportées et la dernière les solutions non supportées. Cette méthode a ensuite été généralisée à k critères [56].

Une méthode séquentielle [114] permet de trouver toutes les solutions non-dominées en ajoutant à chaque nouvelle résolution du problème des contraintes interdisant une solution. Plusieurs variations ont été proposées pour cette méthode [207, 208], permettant notamment de réduire le nombre de contraintes [129].

Il existe également des algorithmes de type *branch-and-bound* multi-critères [37]. Pour rappel, un algorithme de *branch-and-bound* consiste à énumérer des solutions candidates grâce à un arbre de recherche. Chaque branche de l'arbre correspond à un ensemble de solutions et avant d'énumérer toutes les solutions d'une branche, l'algorithme estime les limites inférieures et supérieures des solutions de la branche. Si aucune solutions de la branche ne peut être meilleure que la meilleure solution trouvée jusqu'à présent par l'algorithme, alors cette branche est élaguée. Une des différences en multi-critère est qu'il y a une limite supérieure et inférieure pour chaque critère [37].

Enfin, une méthode, appelée *level set* [71], se base sur des ensembles de solutions, appelés *level set*, déterminés suivant un seuil sur une des fonctions objectifs. Cette méthode repose sur la détermination des k meilleures solutions d'un problème d'optimisation combinatoire et a été appliquée au problème d'affectation quadratique (problème où la fonction objectif est quadratique).

5.1.2 Obtention de courbes de Pareto approchées

Des méthodes permettant d'obtenir des fronts de Pareto approchés (voir définition 1.2.17, page 28) avec garantie de performance ont aussi été développées.

Une manière d'approximer un front de Pareto est de concevoir un algorithme dédié à un problème donné. C'est le cas d'une méthode [223] développée pour les problèmes bi-objectif d'optimisation pour des turbomachines

(compresseurs, pompes, turbines à vapeur et à gaz). Cette méthode est basée sur la méthode des gradients, où il faut calculer les gradients de chaque fonction objectif par rapport aux variables de décision.

Une autre approche consiste à concevoir un algorithme générique. Papadimitriou et Yannakakis [163] ont ainsi montré, pour une large classe de problèmes multi-critères, qu'il existait toujours une courbe de Pareto ϵ-approchée, de taille polynomiale par rapport à $1/\epsilon$, pour tout $\epsilon > 0$, et qui peut être construite en temps polynomial en fonction de la taille de l'instance et de $1/\epsilon$, sous réserve de pouvoir résoudre un problème d'optimisation mono-critère associé.

Il existe également des méta-heuristiques permettant de résoudre les problèmes multi-objectifs de manière approchée sans garantie de performance. On peut citer notamment la recherche locale [201] et la recherche tabou [106] qui, comme vu en section 3.2.5, permettent d'explorer l'espace des solutions du problème en améliorant la solution courante à chaque itération pour la recherche locale, et en interdisant les solutions déjà trouvées pour la recherche tabou.

Le recuit simulé [106], inspiré de l'évolution d'un système physique, est également utilisé. Il s'agit, comme pour la recherche locale, de modifier une solution itérativement pour explorer l'espace des solutions. Cependant, contrairement à la recherche locale, la solution peut être dégradée, avec une certaine probabilité, ce qui permet de s'échapper d'optima locaux.

Les algorithmes génétiques [116] sont également très utilisés. Ce sont des algorithmes bio-inspirés visant à reproduire l'évolution des organismes. Pour cela, à chaque étape, une population de solutions évolue en appliquant des mutations à chaque solution ou des croisements entre des paires de solutions.

De nombreuses métriques existent pour évaluer les méthodes trouvant une courbe de Pareto approchée. Ces métriques évaluent par exemple la distance moyenne entre chaque point de la courbe approchée et de la courbe exacte ou la couverture (nombre de points trouvés) [188]. De par leur diversité et leur nombre, il n'est pas aisé de choisir les bonnes métriques pour déterminer la valeur d'une méthode.

5.1.3 Obtention des k meilleures solutions d'un programme linéaire mono-critère

Il existe quelques méthodes pour trouver les k meilleures solutions d'un programme mono-critère, c'est-à-dire la solution optimale et $k-1$ sous-optimales. Il n'existe en revanche, comme nous l'avons dit en introduction, aucune méthode de ce type en optimisation multi-critère, c'est pourquoi nous désirons développer une méthode générant les k meilleurs ensembles de Pareto.

La première méthode permettant de trouver les k meilleures solutions, développée par Lawler [121], fonctionne en résolvant k fois le problème mono-objectif. À chaque nouvelle résolution, le programme est modifié : une contrainte est ajoutée pour chaque solution déjà trouvée. C'est la contrainte décrite ci-dessous en section 5.2.2. Cette contrainte a été développée par Balas et Jeroslow [14]. Elle ne concerne en revanche que les problèmes à variables de

décision binaires.

Une autre méthode [217], qui fonctionne de la même manière que la première méthode, mais pour des problèmes en nombres entiers, ajoute à chaque nouvelle résolution un ensemble de contraintes, ainsi que des variables continues et des constantes. Ces modifications sont décrites ci-dessous également en section 5.2.2.

Enfin, un nouvel algorithme [87] basé sur un arbre de recherche binaire améliore la méthode de Lawler.

5.2 Notre méthode : Algorithmes et preuves

Dans cette section, nous présentons un nouvel algorithme générique déterminant les k meilleurs ensembles de Pareto d'un PLNE bi-objectif ainsi que la preuve de cet algorithme. Notre algorithme est basé sur la méthode ϵ*-constraint* [21]. Le principe de notre algorithme est d'effectuer une recherche dichotomique. À chaque étape de la recherche, un programme linéaire mono-objectif (une transformation du programme linéaire bi-objectif à résoudre), ayant des contraintes maintenant la deuxième fonction objectif entre deux seuils, est résolu. Si une solution à ce programme est trouvée, l'espace de recherche est découpé en deux à partir de la solution : deux nouvelles recherches sont lancées, en dessous et au dessus de la solution trouvée. La recherche s'arrête dès que toutes les solutions appartenant aux k meilleurs ensembles de Pareto sont trouvées.

Nous présentons dans un premier temps la méthode ϵ*-constraint* sur laquelle est basée notre méthode.

5.2.1 La méthode ϵ-*constraint*

Le principe de cette méthode est de ramener le problème bi-critère à résoudre une suite de problèmes mono-critères. Par simplicité, on suppose que les deux fonctions objectifs doivent être minimisées. Soit BIP (*Bi-objective problem*), le problème bi-objectif général suivant :

$$\min f_1(x)$$
$$\min f_2(x)$$

tel que :

$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$
$$x = (x_1, x_2, \ldots, x_n)$$
$$x_i \in \mathbb{Z} \qquad 1 \leq i \leq n.$$

Les contraintes sont définies ici comme des fonctions linéaires g_j de x.

Pour obtenir un problème mono-critère, une des deux fonctions objectifs est traitée en tant que contrainte : elle doit être inférieure ou égale à un seuil ϵ. On obtient le problème mono-objectif suivant, noté $P(\epsilon)$:

$$\min f_1(x),$$

tel que :

$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$

$$f_2(x) \leq \epsilon$$

$$x_i \in \mathbb{Z} \qquad 1 \leq i \leq n.$$

Nous notons I et N les points idéal et nadir (voir définition 1.2.19, page 28) et $\mathscr{R}$ l'ensemble des solutions du front de Pareto (voir définition 1.2.15, page 27).

L'algorithme 6, ci-dessous, décrit la méthode *ϵ-constraint*. Il consiste tout d'abord à générer les points idéal et nadir pour ajouter la solution $(f_1(I), f_2(N))$ à $\mathscr{R}$. C'est la solution de départ de l'algorithme. À partir de cette solution, le problème $P(\epsilon)$ est résolu en boucle en contraignant la fonction objectif f_2 à être inférieure à $\epsilon = f_2(s) - \delta$, avec s la dernière solution de $P(\epsilon)$ trouvée et δ une constante la plus petite possible. La constante δ permet de trouver des solutions dans une zone de recherche excluant toutes les solutions précédemment trouvées. Elle doit être telle que $\delta < |f_2(s) - f_2(s')|$ pour toute paire de solutions $s \neq s'$ du problème. Cela va restreindre l'espace de recherche et va permettre de trouver les nouvelles solutions appartenant au front de Pareto. Chaque nouvelle solution trouvée s est ajoutée à $\mathscr{R}$. La boucle se termine quand la zone de recherche a été complètement parcourue. Des solutions non-dominées peuvent être trouvées et la dernière étape de l'algorithme consiste à éliminer les solutions non-dominées de $\mathscr{R}$.

Résultat : Front de Pareto exact.
1 Calculer les points idéal et nadir ;
2 $\mathscr{R} := \{(f_1(I), f_2(N))\}$;
3 $\epsilon = f_2(N) - \delta$ $(\delta > 0)$;
4 **tant que** $\epsilon > f_2(I)$ **faire**
5 $\quad s := résoudre(P(\epsilon))$;
6 $\quad \mathscr{R} := \mathscr{R} \cup \{s\}$;
7 $\quad \epsilon = f_2(s) - \delta$;
8 **fin**
9 Supprimer les solutions dominées de $\mathscr{R}$;

Algorithme 6 : La méthode *ϵ-constraint* appliquée à un problème bi-objectif

5.2.2 Notre méthode de résolution d'un problème bi-objectif : FindKBestParetoSets

Dans cette section, nous présentons notre algorithme générique déterminant les k meilleurs ensembles de Pareto d'un programme linéaire en nombres entiers bi-objectif. Cet algorithme est légèrement différent de celui présenté en section 2.2.2. Nous présentons ici la méthode générale avec des améliorations dans l'algorithme. Le programme linéaire bi-objectif que nous voulons résoudre est identique au programme BIP décrit en section 5.2.1.

Par simplicité, nous supposons que les variables de décisions sont binaires. Nous modifions le programme bi-objectif en programme mono-objectif, noté $P(\lambda^-, \lambda^+, F)$, considérant un ensemble de solutions interdites F. Le programme linéaire en nombres entiers $P(\lambda^-, \lambda^+, F)$ est le suivant :

$$\min f_1(x)$$

tel que :

$$f_2(x) \geq \lambda^-$$
$$f_2(x) \leq \lambda^+$$
$$\text{DIFF}(s) \forall s \in F$$
$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$
$$x = (x_1, x_2, \ldots, x_n)$$
$$x_i \in \{0, 1\} \qquad 1 \leq i \leq n.$$

Dans le programme P, la fonction objectif correspond à la première fonction objectif du programme bi-objectif original, et la deuxième fonction objectif est exprimée à travers deux contraintes maintenant sa valeur entre deux seuils : λ^- et λ^+. Nous supposons que ce programme est réalisable et nous notons $\mathscr{X}$ l'ensemble non vide de toutes les solutions possibles.

Pour chaque solution s de F, une contrainte notée $\text{DIFF}(s)$ [14] est ajoutée pour interdire de trouver cette solution. La contrainte est définie de la manière suivante. Supposons que nous avons trouvé précédemment une solution $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$. Cette solution se trouve dans l'ensemble F. La contrainte $\text{DIFF}(x^*)$ est la suivante : $\sum_{i \text{ t.q. } x_i^*=1}(1 - x_i) + \sum_{i \text{ t.q. } x_i^*=0} x_i \geq 1$. Elle assure que la distance de Hamming entre chaque solution possible s et la solution interdite x^* est d'au moins de 1. Ainsi, au moins une variable de décision x_i doit prendre une valeur différente de x_i^*.

Dans le cas d'un programme linéaire à variables de décision entières et non plus seulement binaires, plusieurs variables binaires et continues doivent être ajoutées, ainsi que plusieurs contraintes, résultant en un programme linéaire mixte [217]. Pour chaque solution $x^* = (x_1^*, x_2^*, \ldots, x_n^*) \in F$:

— n variables de décisions binaires sont créées : $\alpha_i \in \{0, 1\}$, $1 \leq i \leq n$,
— n variables de décisions continues sont créées : $W_i \geq 0$, $1 \leq i \leq n$,
— et les contraintes suivantes sont ajoutées au programme linéaire, où M est une grande constante :

$$0 \leq W_i - x_i + x_i^* \leq M(1 - \alpha_i), \quad 1 \leq i \leq n \tag{5.1}$$

$$0 \leq W_i - x_i^* + x_i \leq M\alpha_i, \quad 1 \leq i \leq n \tag{5.2}$$

$$\sum_{i=1}^{n} W_i \geq 1. \tag{5.3}$$

Les contraintes 5.1 et 5.2 servent à définir les variables W_i telles que $W_i = |x_i - x_i^*|$, avec $1 \leq i \leq n$. Une variable W_i représente donc la différence d'une variable i entre les solutions x et x^*. D'après les contraintes 5.3 on a $\sum_{i=1}^{n} W_i = \sum_{i=1}^{n} |x_i - x_i^*| \geq 1$. Ces dernières contraintes permettent donc qu'il y ait au moins une différence entre les variables des deux solutions x et x^*.

Les modifications effectuées dans P pour interdire l'ensemble des solutions de F ne modifient pas la nature du programme qui reste linéaire.

Le principe de l'algorithme que nous proposons est d'effectuer une recherche dichotomique dans l'espace de solutions au programme linéaire P. Pour cela, après chaque nouvelle solution trouvée, l'espace de recherche est divisé en deux.

Soit k le nombre d'ensembles de Pareto recherchés. À la fin de l'algorithme, l'ensemble $\mathscr{R}$ contient toutes les solutions appartenant aux k meilleurs ensembles de Pareto. Chaque solution trouvée par l'algorithme possède un label, noté $l(s)$, indiquant l'index de l'ensemble de Pareto auquel la solution appartient, c'est-à-dire, $l(s) = k$ si et seulement si s appartient au k^{e} ensemble de Pareto. Notre algorithme, dénommé FindKbestParetoSets, fonctionne en appelant la procédure LocalPareto qui effectue la recherche dichotomique de manière récursive.

Résultat : Les k meilleurs ensembles de Pareto exacts
1 $\mathscr{R} := \cup_{i=1}^{k} \mathscr{R}_i$ où $\mathscr{R}_i := Liste[]$;
2 LocalPareto($-\infty$, $+\infty$, $Liste[]$) ; `// Procédure 2`
3 $\mathscr{R} := \mathscr{R} \backslash \{\mathscr{R}_i, i > k\}$;
4 **retourner** $\mathscr{R}$

Algorithme 7 : FindKbestParetoSets

Les ensembles $\mathscr{R}_i$ sont des sous ensembles de $\mathscr{R}$ tels que $\mathscr{R}_i = \{s \in \mathscr{R} | l(s) = i\}$ et $Liste[]$ est une liste vide.

La procédure LocalPareto correspond au calcul d'une portion d'un ensemble de Pareto. Dans cette procédure, l'instruction $ajoute(F, s)$ ajoute la solution s à la fin de la liste F et l'instruction $suppr(F)$ retourne la liste F privée de son dernier élément. Elle est effectuée entre deux seuils pris en entrée, λ^- et λ^+, de la fonction objectif f_2. L'ensemble F représente l'ensemble des solutions trouvées par les appels précédents et dont la valeur de la fonction f_2 se situe entre les seuils λ^- et λ^+. Si le problème $P(\lambda^-, \lambda^+, F)$ a une solution s, le label de s est calculé suivant les lignes 3 à 7. Si le label est inférieur ou égal à $k+1$, la solution s est ajoutée à $\mathscr{R}_{l(s)}$. Si cela est nécessaire, les labels des solutions trouvées précédemment de $\mathscr{R}$ sont mis à jour (lignes 11-13). Enfin, la procédure LocalPareto est appelée pour continuer la recherche en dessous de la solution s (entre λ^- et $f_2(s) - \epsilon$) et au dessus de s (entre $f_2(s)$ et λ^+ si le label de s est inférieur ou égal à k). L'ensemble F est mis à jour automatiquement.

5.2.3 Preuve

Cette section est dédiée à la preuve de l'algorithme 7. Dans la suite, nous notons $\mathscr{X}$ l'ensemble des solutions possibles d'un programme linéaire en nombres entiers bi-objectif, et nous notons $\mathscr{R} \subseteq \mathscr{X}$ un sous-ensemble de solutions. Nous donnons tout d'abord quelques définitions et propriétés nécessaires à la compréhension de la preuve.

Résultat : Une solution de $P(\lambda^-, \lambda^+, F)$

```
1  s := résoudre(P(λ⁻, λ⁺, F)) ;
2  si s ≠ ∅ alors
3      si 𝓛 := {ℛ_i ∈ ℛ, ∃x ∈ ℛ_i t.q. x ≻ s} ≠ ∅ alors
4          l(s) := |𝓛| + 1 ;
5      sinon
6          l(s) := 1 ;
7      fin
8      si l(s) ≤ k + 1 alors
9          ajoute(ℛ_{l(s)}, s) ;
10         si (∃x ∈ ∪_{i=1}^{k} ℛ_i t.q. f1(x) = f1(s) ET x ≠ s) ET
             (∄x ∈ ∪_{i=1}^{k} ℛ_i t.q. f1(x) = f1(s) ET f2(x) = f2(s) ET x ≠ s) alors
11             pour x ∈ ∪_{i=1}^{k} ℛ_i t.q. f1(x) = f1(s) ET x ≺ s faire
12                 ℛ_{l(x)} := suppr(ℛ_{l(x)}) ;
13                 l(x) := l(x) + 1 ;
14                 ajoute(ℛ_{l(x)}, x) ;
15             fin
16         fin
17         LocalPareto(λ⁻, f2(s) − ϵ, F) ;
18         si l(s) ≤ k alors
19             si λ⁻ = f2(s) OU F = Liste[] alors
20                 LocalPareto(f2(s), λ⁺, ajoute(F, s)) ;
21             sinon
22                 LocalPareto(f2(s), λ⁺, ajoute(suppr(F), s)) ;
23             fin
24         fin
25     fin
26 fin
```

Procédure 2 : LocalPareto(λ^-, λ^+, F)

Définition 5.2.1. Une solution $x \in \mathscr{X}$ est *k-Pareto efficace* par rapport aux solutions de $\mathscr{R}$, si et seulement s'il existe $k-1$ solutions $x_i \in \mathscr{R}$, $1 \leq i \leq k-1$, avec $k-1$ maximum, telles que $x_1 \succ \cdots \succ x_{i-1} \succ x_i \succ x_{i+1} \succ \cdots \succ x_{k-1} \succ x$. Si $\nexists x_1$ telle que $x_1 \succ x$, la solution x est dite 1*-Pareto efficace*.

Définition 5.2.2. Le *rang* d'une solution x appartenant à $\mathscr{R}$, noté $rang_{\mathscr{R}}(x)$, est égal à k si et seulement si x est k-Pareto efficace dans $\mathscr{R}$.

D'après les définitions 5.2.1 et 5.2.2, nous pouvons déduire la propriété suivante :

Propriété 5.2.1. Pour toute solution $x \in \mathscr{R}$, si $\{x' \in \mathscr{R}, x \prec x'\} = \emptyset$, alors $rang_{\mathscr{R}}(x) = 1$, sinon $rang_{\mathscr{R}}(x) = max_{\{x' \in \mathscr{R}, x \prec x'\}} rang_{\mathscr{R}}(x') + 1$.

Définition 5.2.3. Soit $\mathscr{X}$ l'ensemble des solutions possibles d'un programme linéaire en nombres entiers bi-objectif. Le *k^e ensemble de Pareto* est défini par $\mathscr{P}_k := \{x \in \mathscr{X} : x$ est k-Pareto efficace dans $\mathscr{X}\}$.

Définition 5.2.4. Soit $\mathscr{R} \subseteq \mathscr{X}$ un sous-ensemble de solutions possibles avec leur labels, d'un programme linéaire en nombres entiers bi-objectif. $\mathscr{R}$ est *cohérent* si et seulement si $l(x) = rang_{\mathscr{R}}(x)$, $\forall x \in \mathscr{R}$.

Définition 5.2.5. Soit $\mathscr{R}$ un ensemble de solutions possibles d'un programme linéaire en nombres entiers bi-objectif qui soit cohérent. Toute solution $x \in \mathscr{R}$ est dite *$\mathscr{R}$-cohérente*.

Définition 5.2.6. Soit $\mathscr{R} \subseteq \mathscr{X}$ un sous-ensemble de solutions possibles avec leur labels, d'un programme linéaire en nombres entiers bi-objectif. $\mathscr{R}$ est *complet* si et seulement si toutes les solutions k-Pareto efficaces dans $\mathscr{X}$ se trouvent dans $\mathscr{R}$.

Les objectifs de cette preuve sont de montrer :
— que l'ensemble $\mathscr{R}$ retourné par l'algorithme 7 est cohérent,
— que l'ensemble $\mathscr{R}$ retourné par l'algorithme 7 est complet,
— et que l'ensemble F possède toujours toutes les solutions qui doivent être interdites.

Nous étudierons également la complexité en temps de l'algorithme.

Notations et vocabulaire Lors d'un appel de la procédure LocalPareto, la solution s calculée en ligne 1 (s'il y a une solution au programme P) est appelée la *solution trouvée* durant cet appel, ou la *solution courante* à cet appel.

Nous notons s_0 la première solution trouvée lors de l'appel LocalPareto$(-\infty, +\infty, \emptyset)$, et par $s_1, s_2, \ldots$, les solutions trouvées par la suite. Nous notons $I_{s_n} := [\lambda^-_{s_n}, \lambda^+_{s_n}]$ l'intervalle dans lequel la solution s_n a été trouvée.

L'exécution de l'algorithme peut être représentée par un arbre (voir définition 1.2.10, page 24), où un nœud est une exécution de la procédure LocalPareto et une branche un appel de la procédure LocalPareto. Ces arbres sont assimilés à des arbres binaires car durant une exécution de la procédure il y a toujours deux appels de la procédure si une solution est trouvée ou zéro appel si aucune solution est trouvée. Lors des dernières exécutions de la procédure, il peut y avoir seulement un appel (si la condition de la procédure LocalPareto à la ligne 18 n'est pas remplie). Cela représente un petit nombre d'exécutions et peut être écarté dans un premier temps. Chaque appel vers la zone sous la solution courante trouvée (ligne 17 de la procédure LocalPareto) est symbolisé par une branche gauche et un appel au dessus de la solution courante trouvée (lignes 20 ou 22) est symbolisé par une branche droite. Un exemple est visible en figure 5.1.

Étant donné une solution s dans cet arbre, nous notons $p(s)$ sa solution parente. Si s est obtenue à partir d'une branche gauche (et respectivement d'une branche droite) alors $p(s)$ est appelé parent gauche (et respectivement parent droit).

D'après les lignes 17, 20 et 22 de la procédure LocalPareto, nous pouvons déduire la propriété suivante sur les valeurs des seuils λ^- et λ^+ :

Propriété 5.2.2. Soit s_n la n^e solution trouvée durant un appel de la procédure LocalPareto. Soit $p(s_n)$ le parent de s_n. Dans le cas où $p(s_n)$ est un parent gauche, on a $\lambda^-_{s_n} < f_2(p(s_n))$ (figure 5.2 A). Dans le cas où $p(s_n)$ est un parent droit, on a $\lambda^-_{s_n} = f_2(p(s_n))$ (figure 5.2 B).

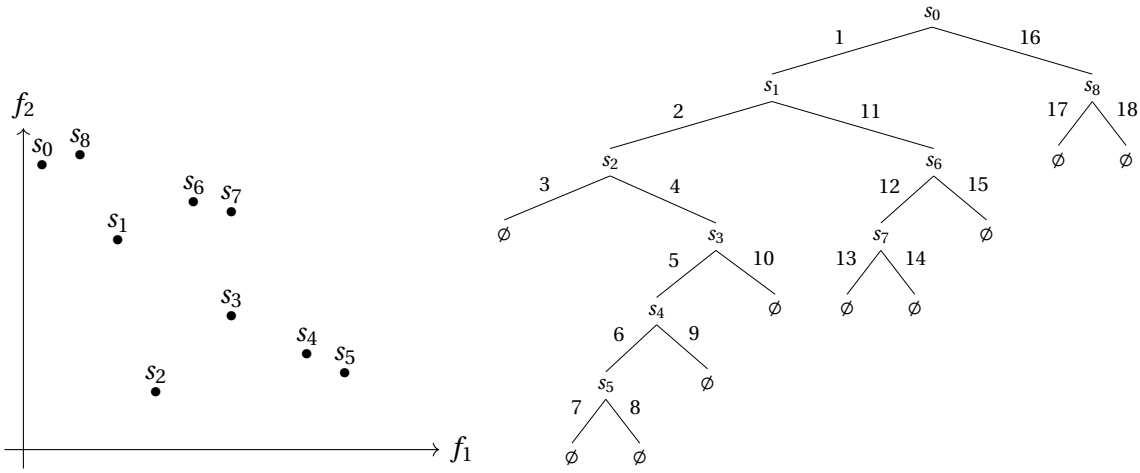

Figure 5.1 : Représentation par un arbre d'un exemple d'exécution de l'algorithme FindKbestParetoSets. À gauche, les solutions sont représentées selon leur valeur pour les objectifs f_1 et f_2. À droite, l'arbre représentant l'exécution : à chaque sommet est associé la solution trouvée durant l'appel et le poids des arêtes représente l'ordre d'exécution des appels.

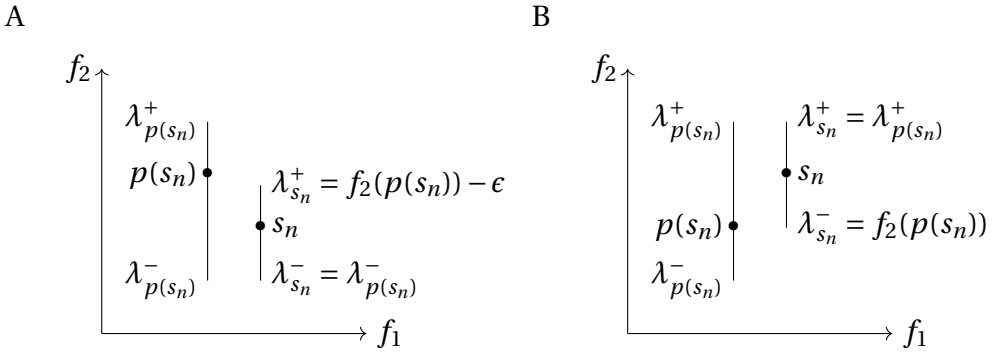

Figure 5.2 : Parentalité entre les solutions de l'algorithme FindKbestParetoSets. A) Quand $p(s_n)$ est un parent gauche de s_n. B) Quand $p(s_n)$ est un parent droit de s_n.

L'ensemble $\mathscr{R}$ est cohérent

Nous cherchons ici à prouver que l'ensemble $\mathscr{R}$ retourné par la procédure LocalPareto est cohérent.

Tout d'abord, nous pouvons déduire, grâce au programme $P(\lambda^-, \lambda^+, F)$ et aux lignes 17, 20 et 22 de la procédure LocalPareto, les propriétés suivantes sur les valeurs prises par les solutions trouvées lors d'un appel de la procédure.

Propriété 5.2.3. Soit s une solution trouvée lors de la procédure LocalPareto et $p(s)$ sa solution parente. On a $f_1(s) \geq f_1(p(s))$.

Propriété 5.2.4. Soit s une solution trouvée lors d'un appel de la procédure LocalPareto$(\lambda^-, f_2(p(s)) - \epsilon, F)$ et $p(s)$ sa solution parente. La solution s est telle que $\lambda^- \leq f_2(s) < f_2(p(s))$.

Propriété 5.2.5. Soit s une solution trouvée lors d'un appel de la procédure LocalPareto$(f_2(s), \lambda^+, F)$ et $p(s)$ sa solution parente. La solution s est telle que $f_2(p(s)) \leq f_2(s) \leq \lambda^+$.

Les solutions se trouvant dans l'ensemble $\mathscr{R}$ sont donc telles que :

Propriété 5.2.6. Considérons tout appel de la procédure LocalPareto$(\lambda^-, \lambda^+, F)$. Soit s une solution trouvée lors de cet appel. Toute solution $x \in \mathscr{R}$ lors de cet appel satisfait :

— si $\lambda^- = -\infty$: $f_2(x) \geq f_2(p(s))$ et $f_1(x) \leq f_1(s)$ (figure 5.3 A),

— si $\lambda^- \neq -\infty$: ($f_2(x) \in]-\infty, +\infty[$ et $f_1(x) \leq f_1(p(s))$) ou ($f_2(x) \leq \lambda^-$ et $f_1(x) \in]-\infty, +\infty[$) (figures 5.3 B et C).

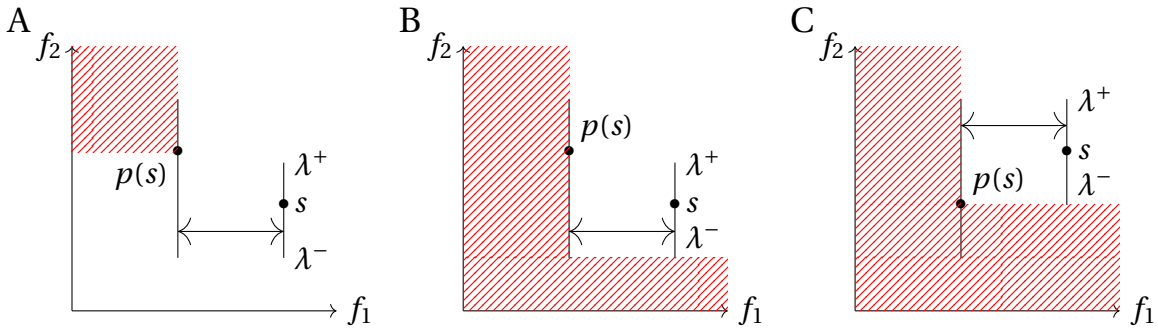

Figure 5.3 : Illustration de la propriété 5.2.6. Les zones hachurées correspondent aux zones où les solutions de l'ensemble $\mathscr{R}$ peuvent se trouver. La double flèche indique la longueur entre les solutions s et $p(s)$, cette longueur peut être nulle. A) Quand $\lambda^- = -\infty$. B) Quand $\lambda^- \neq -\infty$ et que $p(s)$ est un parent gauche. C) Quand $\lambda^- \neq -\infty$ et que $p(s)$ est un parent droit.

Montrons par récurrence que l'ensemble $\mathscr{R}$ est cohérent.

Démonstration. Initialement, l'ensemble $\mathscr{R}$ est vide, il est donc cohérent.

Nous supposons qu'après n appels de la procédure LocalPareto, $\mathscr{R}$ est cohérent.

Montrons qu'après l'appel $n+1$, $\mathscr{R}$ est toujours cohérent.

D'après la propriété 5.2.6 nous pouvons voir que suivant la position des solutions dans $\mathscr{R}$, le label de la solution courante s va être influencé.

Les lignes 3 à 7 de la procédure LocalPareto permettent de déterminer le label de la solution s. Si s est non-dominé, son label est égal à 1, sinon il est égal à la taille de $\mathscr{L}$. D'après la définition des $\mathscr{R}_i$, la taille de l'ensemble $\mathscr{L}$ correspond à $max_{\{x' \in \mathscr{R}, x < x'\}}$. Donc le label de s correspond à son rang dans $\mathscr{R}$, $rang_{\mathscr{R}}(s) = l(s)$ (propriété 5.2.1). À ce moment, la solution s n'est pas encore ajoutée à l'ensemble $\mathscr{R}$, donc ce dernier est toujours cohérent.

Si la solution est ajoutée (ligne 9), l'ensemble $\mathscr{R}$ peut ne plus être cohérent car la solution s peut être alignée verticalement avec d'autres solutions de $\mathscr{R}$, ce qui fait que leur labels ne correspondront plus à leur rang (figure 5.4). C'est le seul cas où le label des autres solutions peut être impacté (voir figure 5.3 : lorsque la distance entre $p(s)$ et s est nulle, des solutions de $\mathscr{R}$ peuvent être alignées verticalement avec s et avoir une valeur f_2 supérieure). Ce cas

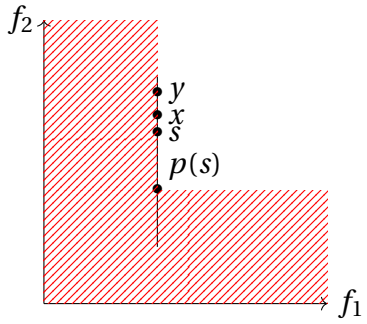

Figure 5.4 : Cas d'un alignement vertical de solutions. Les solutions x et y sont des solutions trouvées avant la solution $p(s)$. Les zones hachurées correspondent aux zones où les solutions de l'ensemble $\mathscr{R}$ peuvent se trouver.

nécessite une mise à jour des labels des solutions $x \in \mathscr{R}$ telles que $f_1(x) = f_1(s)$ et $f_2(x) > f_2(s)$ (première partie de la condition Si de la ligne 10). En revanche, s'il existe une solution $x \in \mathscr{R}$ ayant les même valeurs objectifs que s, la mise à jour aura déjà été faite et donc $\mathscr{R}$ sera toujours cohérent (seconde partie de la condition Si de la ligne 10).

D'après la définition 5.2.1 et la propriété 5.2.1, les labels des solutions nécessitant la mise à jour doivent être incrémentés de 1 pour correspondre à leur rangs. C'est ce qui est effectué dans la procédure aux lignes 11 à 15. L'ensemble $\mathscr{R}$ est alors de nouveau cohérent (définition 5.2.4).

L'ensemble n'est pas modifié dans la suite de la procédure donc l'ensemble $\mathscr{R}$ après $n+1$ appels de la procédure LocalPareto est cohérent.

Par le principe de récurrence, nous montrons que l'ensemble $\mathscr{R}$ est cohérent. □

L'ensemble $\mathscr{R}$ est complet

Nous cherchons ici à prouver que l'ensemble $\mathscr{R}$ retourné par l'algorithme 7 est complet, pour k ensembles de Pareto recherchés.

Démonstration. Nous supposons qu'il existe $\epsilon > 0$ telle que pour toute paire de solutions s et s', on a soit $f_2(s) = f_2(s')$ ou $|f_2(s) - f_2(s')| > \epsilon$. La procédure LocalPareto commence par faire une recherche dans l'intervalle $I_{s_0} := [-\infty, +\infty]$ de f_2. Elle procède à un appel récursif (lignes 17, 22 et 20) à partir de toute nouvelle solution trouvée, ce qui permet de trouver toutes les solutions dans cet intervalle.

Toutes les valeurs de f_1 pour les solutions possibles sont également parcourues. La zone de recherche des valeurs de f_1 débute avec la valeur optimale de f_1 quand la première solution est trouvée (elle optimise seulement le critère f_1). À chaque appel de la procédure LocalPareto, le programme linéaire P est résolu, donc pour toute nouvelle solution trouvée s, on a de fait $f_1(s) \geq f_1(p(s))$.

Les appels vers le bas s'arrêtent jusqu'aux solutions ayant des labels au plus égaux à $k+1$ (ligne 8). En effet, lorsqu'une solution $p(s)$ a un label tel que

$l(p(s)) = k+1$, la solution s trouvée peut être alignée verticalement avec $p(s)$ et donc avoir un label tel que $k \leq l(s) \leq k+1$ (voir figure 5.3 B).

Les appels vers le haut s'arrêtent jusqu'aux solutions ayant des labels au plus égaux à k (ligne 18). En effet, si une solution $p(s)$ avait un label tel que $l(p(s) = k+1$ lors de ces appels, la future solution trouvée s aurait un label tel que $l(s) \geq l(p(s)) = k+1$ (voir figure 5.3 C).

Comme nous venons de le voir, à chaque appel de la procédure LocalPareto, une solution est ajoutée seulement si son label est inférieur ou égal à $k+1$. De plus, à la fin de l'algorithme FindKbestParetoSets, seules les solutions ayant un label inférieur ou égal à k sont conservées dans $\mathscr{R}$ (ligne 3). Enfin, comme nous l'avons montré précédemment, l'ensemble $\mathscr{R}$ est cohérent, donc toutes les solutions ajoutées correspondent aux k ensembles de Pareto du programme linéaire bi-objectif. □

L'ensemble F est mis à jour automatiquement

Nous cherchons ici à montrer par récurrence que l'ensemble F contient toujours les solutions qui doivent être interdites lors d'un appel de la procédure LocalPareto, c'est-à-dire, pour tout appel LocalPareto$(\lambda^-, \lambda^+, F)$, F contient exactement les solutions $x \in \mathscr{R}$ telles que $\lambda^- \leq f_2(x) \leq \lambda^+$.

Démonstration. Lors du premier appel de la procédure LocalPareto, l'ensemble F est vide. Comme aucune solution n'a encore été trouvée, il n'y a pas besoin d'interdire de solutions.

Nous supposons qu'au début de l'appel n, F contient exactement les solutions $x \in \mathscr{R}$ qui doivent être interdites à l'appel n, elles sont telles que $\lambda^-_{s_n} \leq f_2(x) \leq \lambda^+_{s_n}$.

Montrons qu'à la fin de l'appel n, F contient exactement les solutions qui doivent être interdites dans les deux prochains appels.

Pour effectuer les prochains appels, nous pouvons ajouter ou supprimer des solutions dans F. La solution s_n est une solution potentielle à interdire et donc pourrait être ajoutée à F. Des solutions de F peuvent également être en dehors des futures zones de recherche et pourront être supprimées de F.

Nous distinguons deux cas :

— Si s_n est un parent droit de la future solution, notée $e(s_n)$: il s'agit d'un appel vers le haut. Les appels vers le haut vont nécessiter l'ajout de la solution s_n car elle est incluse dans la future zone de recherche ; on a ainsi $I_{e(s_n)} = [f_2(s_n), \lambda^+_{e(s_n)}]$ (figure 5.3 C). Cette solution $p(s_n)$ est ajoutée lors des deux appels vers le haut dans la procédure LocalPareto aux lignes 20 et 22. C'est aussi l'unique fois où une solution est ajoutée à F. De plus, d'après la supposition de la récurrence, F contient déjà toutes les autres solutions nécessaires.

  Avant d'effectuer un appel vers le haut, nous pouvons supprimer la dernière solution de F dans certains cas. Nous allons montrer que dans ces cas cette solution est inutile et peut donc être supprimée :

  — Dans la plupart des cas, la solution s_n est telle que $f_2(s_n) \neq \lambda^-_{s_n}$, et en supposant qu'il y ait eu plus de 2 appels de la procédure (F est non vide), alors la dernière solution de F est telle que $f_2(dernier(F)) \notin$

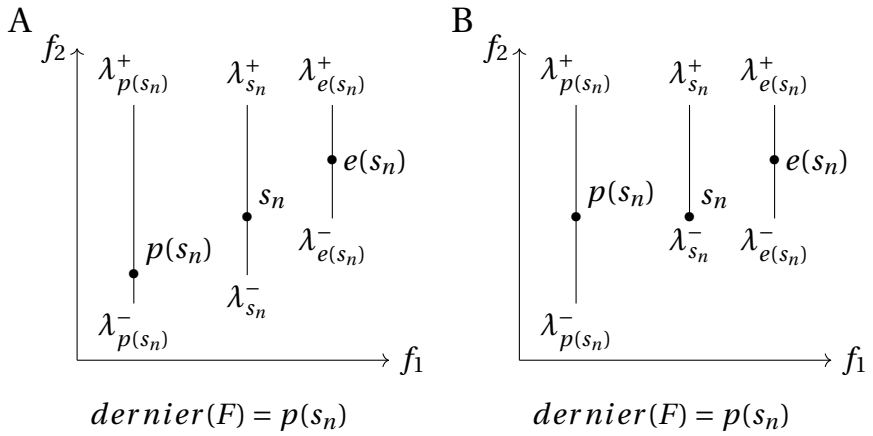

Figure 5.5 : Mise à jour de l'ensemble F lors d'un appel vers le haut. L'enfant de la solution s_n est noté $e(s_n)$. A) Cas où la solution $dernier(F)$ peut être supprimée, $f_2(s_n) \neq \lambda^-_{s_n}$. B) Cas où la solution $dernier(F)$ ne peut pas être supprimée, $f_2(s_n) = \lambda^-_{s_n}$.

$I_{e(s_n)}$ (figure 5.5 A). L'instruction $dernier(F)$ renvoie le élément ajouté à F. Cette solution n'est donc pas nécessaire et peut être supprimée de F ; c'est ce qui est réalisé à la ligne 22 de la procédure car les conditions décrites précédemment sont remplies (les conditions à la ligne 19 ne sont pas remplies donc c'est la ligne 22 qui est exécutée).

— En revanche, si la solution s_n est telle que $f_2(s_n) = \lambda^-_{s_n}$ alors on ne peut pas supprimer la dernière solution de F (si $F = \emptyset$, il n'y a rien à supprimer non plus). En effet, si $f_2(s_n) = \lambda^-_{p(s_n)}$, on sait que $f_2(s_n) = f_2(p(s_n))$ et que $p(s_n) = dernier(F)$, puisqu'elle était issue d'un appel vers le haut (figure 5.5 B). Par conséquent, on a $I_{e(s_n)} = [f_2(p(s_n)), \lambda^+_{s_n}]$, donc il ne faut pas supprimer la solution $p(s_n)$ de F.

Donc, pour un prochain appel vers le haut, F contient exactement les solutions à interdire pour trouver $e(s_n)$.

— Si s_n est un parent gauche de $e(s_n)$: il s'agit d'un appel vers le bas.
  — Lors d'un appel vers le bas (figures 5.3 A et B), la solution s_n sera exclue de la zone de recherche $I_{e(s_n)} = [\lambda^-_{s_n}, f_2(s_n) - \epsilon]$. Il n'est donc pas nécessaire d'ajouter cette solution à F.
  — La dernière solution de F provient d'un appel vers le haut et si cette solution est dans F c'est qu'elle n'a pas été supprimée comme vu précédemment. Elle était donc nécessaire à la recherche de la solution s_n. Comme $I_{e(s_n)} \subseteq I_{s_n}$ et $\lambda^-_{e(s_n)} = \lambda^-_{s_n}$, la dernière solution se trouvera dans l'intervalle $I_{e(s_n)}$, il faut donc la conserver.

  Pour le prochain appel vers le bas, nous avons montré que F contient exactement les solutions à interdire pour trouver $e(s_n)$.

Par le principe de récurrence, nous avons montré que F contenait exactement les solutions à interdire pour tout appel de la procédure F. □

Complexité de l'algorithme

Proposition 1. *La complexité en temps de l'algorithme FindKbestParetoSets (algorithme 7) est* $\mathcal{O}(|\mathcal{S}|^2 + |\mathcal{S}| \times \mathcal{O}(P))$ *avec* $|\mathcal{S}|$ *le nombre de solutions trouvées par l'algorithme, incluant les solutions n'appartenant pas aux k meilleurs ensembles de Pareto, tel que* $|\mathcal{S}| \leq 2 \times |\mathcal{R}|$.

Pour pouvoir borner $|\mathcal{S}|$, nous considérons ici la version du programme $P(\lambda^-, \lambda^+, F)$, résolu dans la procédure LocalPareto, suivante :

$$\min f_1(x) + \varepsilon f_2(x)$$

tel que :

$$f_2(x) \geq \lambda^-$$

$$f_2(x) \leq \lambda^+$$

$$\text{DIFF}(s) \forall s \in F$$

$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$

$$x = (x_1, x_2, \ldots, x_n)$$

$$x_i \in \{0, 1\} \qquad 1 \leq i \leq n$$

où ε est une petite constante pour permettre de minimiser la fonction f_2 sans interférer avec la minimisation de f_1. Utiliser cette version du programme linéaire ne modifie pas l'algorithme.

Démonstration. La complexité en temps de l'algorithme 7 dépend du nombre d'appels de la procédure LocalPareto (ligne 2) qui correspond au nombre de solutions trouvées lors de l'exécution, noté $\mathcal{S}$. L'ensemble $\mathcal{S}$ contient toutes les solutions de $\mathcal{R}$ ainsi que des solutions n'appartenant pas aux k meilleurs ensembles de Pareto recherchés mais qui sont nécessaires à l'exécution de l'algorithme. Nous notons l'ensemble de ces dernières solutions $\mathcal{E}$, ainsi on a : $|\mathcal{S}| = |\mathcal{R}| + |\mathcal{E}|$. Le nombre de solutions appartenant à $\mathcal{E}$ correspond :

— Au nombre de feuilles de l'arbre d'exécution de l'algorithme (au plus $|\mathcal{R}| + 1$). Lorsqu'une solution trouvée lors de la procédure LocalPareto a un label supérieur à $k+1$, la recherche s'arrête (ligne 8) ; cela correspond à une feuille dans l'arbre. Lors d'un appel droit (vers le haut), si la solution trouvée a un label supérieur à k, la recherche s'arrête (ligne 18) ; cela correspond également à une feuille dans l'arbre.
Pour déterminer le nombre de feuilles d'un arbre d'exécution, nous notons n_0, n_1 et n_2, le nombres de sommets ayant respectivement 0, 1 et 2 enfants. Le nombre de solutions de $\mathcal{R}$ est égal à n_2 (si moins de deux appels sont réalisés à partir d'une solution c'est que son label est supérieur à k et ne fait donc pas partie de $\mathcal{R}$). On note $|E|$ le nombre d'arêtes total d'un arbre d'exécution et $|V|$ le nombre total de sommets. On sait que le nombre d'arêtes total d'un arbre est égal au nombre de sommets moins 1. On a donc :

$$|E| = |V| - 1, \quad |E| = 2n_2 + n_1 \quad \text{et} \quad |V| = n_2 + n_1 + n_0,$$

$$n_2 + n_1 + n_0 - 1 = 2n_2 + n_1 \quad \Leftrightarrow \quad n_2 + n_0 - 1 = 2n_2 \quad \Leftrightarrow \quad n_0 = n_2 + 1$$
$$\Rightarrow n_0 = |\mathscr{R}| + 1.$$

Le nombre de feuilles est donc égal à $|\mathscr{R}| + 1$.

— Plus au nombre de nœuds dans l'arbre d'exécution correspondant à des solutions alignées verticalement (au plus $|\mathscr{R}|$). Lorsqu'une solution trouvée lors de la procédure LocalPareto a un label égal à $k+1$, un appel gauche (vers le bas) peut être lancé et il y a une possibilité d'obtenir lors de cet appel une solution ayant un label égal à k et donc que la recherche continue. La solution initiale ayant un label égal à $k+1$ est une solution appartenant à $\mathscr{E}$, mais pas la suivante. Comme le programme linéaire que nous résolvons minimise également f_2, la solution trouvée lors de l'appel gauche suivant est forcément celle ayant la meilleure valeur f_2 et donc il y a dans ce cas, une solution appartenant à $\mathscr{E}$ générée pour une solution appartenant à $\mathscr{R}$.

On a donc $|\mathscr{E}| \leq 2 \times |\mathscr{R}| + 1$.

$\mathscr{R}$ est un vecteur en deux dimensions stockant les solutions trouvées par la procédure. Les solutions y sont ordonnées selon leur label en sous-vecteur : le sous-vecteur correspondant à l'ensemble de Pareto i est noté $\mathscr{R}_i$. Dans chaque sous-vecteur $\mathscr{R}_i$, les solutions sont ordonnées dans l'ordre croissant de leur valeur f_1. L'étape 3 consiste à supprimer les solutions dominées, ce qui peut être effectué en une seule fois en supprimant de $\mathscr{R}$ les ensembles de Pareto tel que $i > k$, où k est le nombre d'ensembles de Pareto recherchés.

La complexité de la procédure LocalPareto dépend de la complexité à résoudre le programme linéaire P, notée $\mathscr{O}(P)$, elle-même dépendant du nombre de variables et de contraintes de P. Hormis la complexité à résoudre P, la complexité de la procédure dépend aussi de la détermination du label de la solution s (lignes 3-7). Les fonctions $ajoute$ et $suppr$ se font en temps constant car l'ensemble $\mathscr{R}$ est une liste chainée. L'étape à la ligne 3 requière uniquement de tester la première solution de chaque ensemble de Pareto ($\mathscr{O}(k)$) pour construire l'ensemble $\mathscr{L}$ et l'étape à la ligne 4 requière uniquement de prendre une solution parmi le dernier ensemble de Pareto.

Après avoir ajouté la solution s à l'ensemble $\mathscr{R}$, la procédure passe ensuite à une étape de mise à jour des labels si nécessaire (lignes 10 à 16). L'étape à la ligne 10 nécessite de chercher des solutions dans l'ensemble $\mathscr{R}$. Cela peut être effectué grâce à une recherche dichotomique sur chaque ensemble de Pareto $\mathscr{R}_i$ car les solutions y sont triés selon leur valeur f_1. Cela est réalisé en $\mathscr{O}(\sum_{i=1}^{k} \log(|\mathscr{R}_i|)$. Ensuite l'étape à la ligne 11 a une complexité de $\mathscr{O}(|\mathscr{R}| - 1)$.

En résumé, la complexité de la procédure LocalPareto est $\mathscr{O}(P) + \mathscr{O}(k) + \mathscr{O}(\sum_{i=1}^{k} \log(|\mathscr{R}_i|) + \mathscr{O}(|\mathscr{R}| - 1)$, *i.e.*, $\mathscr{O}(P) + \mathscr{O}(|\mathscr{R}| - 1)$.

Donc, la complexité en temps de l'algorithme 7 est $O(|\mathscr{S}|) \times [\mathscr{O}(P) + \mathscr{O}(|\mathscr{S}|)]$, *i.e.*, $\mathscr{O}(|\mathscr{S}|^2 + |\mathscr{S}| \times \mathscr{O}(P))$. □

5.2.4 Parallélisation de l'algorithme

Dans cette section nous présentons comment l'algorithme peut être parallélisé et nous montrons que le temps d'exécution peut être réduit d'un facteur 3/4 en moyenne dans les meilleurs cas.

Nous rappelons que l'exécution de l'algorithme peut être représentée par un arbre où chaque nœud correspond à une exécution de la procédure LocalPareto, et où chaque arête correspond à un appel de la procédure. L'arbre est parcouru en profondeur, mais quand plusieurs nœuds sont à la même profondeur, nous commençons toujours par les nœuds les plus à gauche.

Dans la procédure LocalPareto, c'est la résolution du programme linéaire que nous allons paralléliser. En effet, c'est la résolution des programmes linéaires qui nécessite une plus grande complexité en temps. De plus, pour conserver la cohérence de l'ensemble $\mathscr{R}$, la détermination du label de la solution s et la mise à jour des labels de l'ensemble $\mathscr{R}$ doivent être effectués dans le même ordre que si l'algorithme était séquentiel.

Chaque appel d'une branche gauche nécessite le résultat du nœud parent, donc l'exécution des nœuds issus de branches gauches peut seulement être faite de manière séquentielle. Cependant, lorsque l'on parcourt plusieurs branches gauches consécutives, toutes les informations nécessaires aux appels des branches droites sont réunies. Les exécutions pouvant être parallélisées sont celles correspondant à l'ensemble des nœuds issus des branches droites provenant d'un ensemble de branches gauches consécutives. Par conséquent, l'exécution des nœuds issus de ces branches droites peut être faite en parallèle. Une illustration de la parallélisation de l'algorithme sur un exemple est donnée en figure 5.6, en supposant que la résolution de chaque programme linéaire prenne le même temps unitaire.

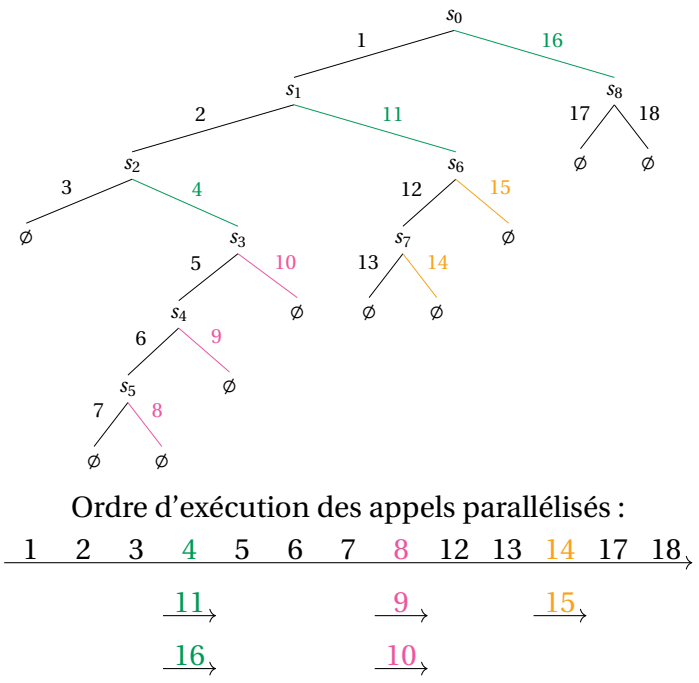

Figure 5.6 : Illustration d'une exécution de l'algorithme FindKbestParetoSets (algorithme 7) parallélisé.

Pour évaluer l'accélération du temps d'exécution de l'algorithme parallélisé, nous définissons le coefficient d'accélération comme le ratio entre le temps

mis par la version parallélisée et le temps mis par la version séquentielle. Dans l'exemple de la figure 5.6, le coefficient d'accélération est égal à 13/18.

Nous allons évaluer ce coefficient sur une famille d'arbres générés aléatoirement. Dans un premier temps, nous considérons une famille d'arbres binaires aléatoires, correspondant aux meilleurs cas pour la parallélisation de l'algorithme, puis nous analysons une famille d'arbres aléatoires avec un modèle de génération plus réaliste.

Meilleurs cas : les arbres binaires

Les arbres correspondant aux meilleurs ratios de parallélisation sont des arbres binaires. Nous les générons de la manière suivante. Nous commençons avec un arbre à trois nœuds : une racine et deux branches. Pour générer la suite de l'arbre, nous procédons en plusieurs étapes. Durant chaque étape, nous ajoutons un nœud suivi de deux branches, soit sur une branche gauche, soit sur une branche droite. La figure 5.7 illustre la génération d'un tel arbre.

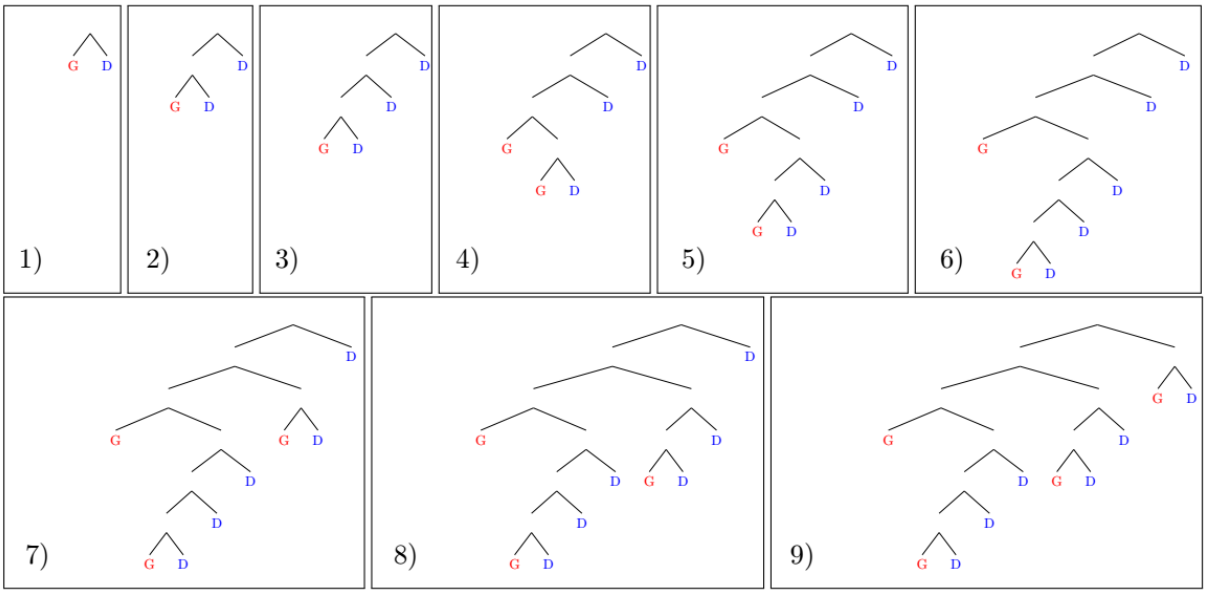

Figure 5.7 : Les différentes étapes de génération d'un arbre binaire aléatoire. G : nœud issu d'une branche gauche. D : nœud issu d'une branche droite.

À partir de la génération d'un arbre, nous pouvons calculer le nombre d'appels qui pourront être parallélisés, c'est ce que nous appelons le gain. Le gain est égal à la différence entre le nombre d'appels de la version séquentielle et le nombre d'appels de la version parallélisée de l'algorithme.

Quand un nouveau nœud est ajouté à une branche gauche, la nouvelle branche droite r qui est ajoutée est parallèle à la branche droite du nœud parent. Ainsi l'appel de la nouvelle branche droite peut être parallélisé avec celui de la branche droite du nœud parent, le gain est incrémenté de un. En revanche, lorsqu'un nœud est ajouté à une branche droite, le nœud parent ne possède pas de branche droite parallèle à la nouvelle branche droite, et donc, cet appel ne peut être parallélisé.

Pour calculer le gain moyen, les ajouts d'un nœud et deux branches lors de la génération d'un arbre sont assimilés à un tirage d'une balle dans un sac, avec remise. La balle peut être rouge ou bleue, la couleur correspondant à l'addition d'un nœud (et de deux branches) soit sur une branche gauche, soit sur une branche droite, respectivement. Quand une balle est tirée, elle est remise avec une balle supplémentaire de l'autre couleur. Ainsi, la probabilité de tirer une balle rouge, ou bleue, correspond à la probabilité d'ajouter un nœud (et deux branches) sur une branche gauche, ou droite respectivement.

Pour calculer le nombre moyen de nœuds parallélisés, nous suivons la règle suivante : quand une balle tirée est rouge, le gain est incrémenté de un, sinon le gain reste le même. En effet, lorsqu'un nœud et deux branches sont ajoutés à une branche gauche, une nouvelle branche droite est ajoutée à un groupe de branches gauches consécutives, ce qui correspond à une exécution pouvant être parallélisée. Quand un nœud et deux branches sont ajoutés à une branche droite, il n'existe pas de groupe de branches gauches parentes consécutives.

Nous pouvons montrer que le gain moyen est égal à $n/2$, avec n le nombre d'étapes pour générer l'arbre. En effet, considérons deux variables aléatoires g et $\overline{g}$, telles que g (et respectivement $\overline{g}$) augmente de 1 à chaque fois que le gain augmente de 1 (et respectivement reste identique). Nous avons pour les espérances de chacune de ces variables aléatoires :

$$E(g) + E(\overline{g}) = n,$$

et par symétrie du processus :

$$E(g) = E(\overline{g}),$$

donc :

$$E(g) = \frac{n}{2}.$$

Le nombre d'appels est égal au nombre de branches de l'arbre : $2n+2$. On a :

$$\text{Coefficient d'accélération} = \frac{\text{temps parallélisé}}{\text{temps séquentiel}} = \frac{2n+2-\frac{n}{2}}{2n+2} = \frac{\frac{3}{2}n+2}{2n+2}$$

Ainsi, en négligeant les +2 on obtient :

$$\text{Coefficient d'accélération} \simeq \frac{3}{4}.$$

Parmi ces arbres binaires, nous pouvons constater qu'un type d'arbre permet de maximiser le coefficient d'accélération. Ce sont des arbres binaires où chaque nœud possède des petits-enfants, c'est-à-dire qu'il y a deux générations d'enfants après lui. En effet, si après un nœud il n'y a qu'une génération d'enfants, le gain est nul. Si le nœud s_9 de l'arbre en figure 5.6 est supprimé, il correspond à ce type d'arbre. Le coefficient d'accélération devient alors 8/16.

Cas moyens

Nous abordons ici la génération d'arbres correspondant exactement à l'exécution de notre algorithme. En effet, précédemment nous avons considéré

uniquement des arbres binaires, mais les appels droits ne sont pas toujours réalisés, dans une proportion dépendant du programme P. Cela est dû à une condition de la procédure LocalPareto (ligne 18) autorisant les appels droits seulement si la solution s trouvée lors de cette exécution a un label inférieur ou égal au nombre d'ensembles de Pareto recherchés.

Pour suivre les états du sac de balles, nous notons $E_n(G, D, g, b)$ l'état du sac à l'étape n de la génération de l'arbre, où G est le nombre de balles rouges (correspondant au nombre de branches gauches auxquelles on peut ajouter des branches), D est le nombre de balles bleues (le nombre de branches droites auxquelles on peut ajouter des branches), g est le gain et b est le nombre de branches.

La génération d'un arbre est similaire à celle de la section précédente. Elle est réalisée grâce au tirage de balles bleues et rouges dans un sac. Une branche gauche correspond à un appel vers le bas, et une branche droite correspond à un appel vers le haut. Une différence est que lorsque nous ajoutons une branche, nous ne savons pas encore s'il y aura une solution trouvée. De plus, nous avons ici des règles supplémentaires lorsque nous devons ajouter une nouvelle branche. Il y a trois possibilités, dont les probabilités dépendent directement de l'espace des solutions du programme linéaire P :

— Événement A : avec une probabilité p, on trouve une solution et les deux prochains appels sont effectués : nous ajoutons deux branches. La balle tirée est remise et une balle de couleur opposée est ajoutée au sac.
— Événement B : avec une probabilité q, on trouve une solution et seul l'appel vers le bas est effectué : seulement une branche gauche est ajoutée. Une balle rouge est mise dans le sac.
— Événement C : avec une probabilité $1 - p - q$, on ne trouve pas de solution : aucune branche n'est ajoutée. La balle tirée n'est pas remise dans le sac.

Ces possibilités et les calculs du gain et du nombre de branches en fonction de ces possibilités sont résumées dans le tableau 5.1. Le tirage de balles est réalisé tant que le sac n'est pas vide.

Le calcul du gain et le nombre de branches diffèrent suivant quel événement est réalisé. Nous pouvons calculer le gain moyen, noté $\widehat{g}$, ainsi que le nombre de branches moyen, noté $\widehat{b}$, grâce aux équations suivantes :

$$\widehat{g} = N \times (p + q), \tag{5.4}$$

$$\widehat{b} = N \times [2(p + q) + (1 - p - q - m + m)] + 2 \iff N(p + q) + N + 2, \tag{5.5}$$

où N est le nombre d'étapes de génération d'un arbre.

En revanche, nous ne pouvons calculer le coefficient d'accélération moyen car $(1 - \widehat{g})/\widehat{b}$ est le produit de deux espérances et ne correspond pas à l'espérance du coefficient d'accélération.

Pour estimer le coefficient d'accélération nous avons effectué quelques tests. Nous avons généré 2000 arbres d'exécutions (100 étapes maximum pour générer un arbre) et avons calculé pour chaque arbre le coefficient d'accélération. Par exemple pour $p = 0.9$ et $q = 0.05$, le coefficient moyen est de 0.765.

Nous avons ensuite étudié l'évolution du coefficient d'accélération en fonction de la probabilité p (figure 5.8). Par exemple, pour $p = 1$, le coefficient d'ac-

Événement	Balle tirée	Probabilité	Modification de l'arbre	Mise à jour du sac de balles	E_{n+1}
A	Rouge	$p \times \frac{G}{G+D}$	G G D	La balle rouge est remise et une balle bleue est ajoutée au sac.	$E_{n+1}(G, D+1, g+1, b+2)$
A	Bleue	$p \times \frac{D}{G+D}$	D G D	La balle bleue est remise et une balle rouge est ajoutée au sac.	$E_{n+1}(G+1, D, g, b+2)$
B	Rouge	$q \times \frac{G}{G+D}$	G G	Une balle rouge est mise dans le sac.	$E_{n+1}(G, D, g, b+1)$
B	Bleue	$q \times \frac{D}{G+D}$	D G	Une balle rouge est mise dans le sac.	$E_{n+1}(G+1, D-1, g, b+1)$
C	Rouge	$(1-p-q) \times \frac{G}{G+D}$	R $\rightarrow \emptyset$	Aucune balle n'est mise ou remise dans le sac.	$E_{n+1}(G-1, D, g, b+1)$
C	Bleue	$(1-p-q) \times \frac{D}{G+D}$	D $\rightarrow \emptyset$	Aucune balle n'est mise ou remise dans le sac.	$E_{n+1}(G, D-1, g, b+1)$
Le sac est vide.	-	P(Le sac est vide)	-	Aucune balle n'est mise dans le sac.	$E_{n+1}(G, D, g, b)$

Tableau 5.1 : Règles lors de la génération d'un arbre représentant l'exécution de l'algorithme FindKbestParetoSets.

célération est égal à 0.74, ce qui signifie que l'exécution parallèle a en moyenne une durée égale à 0.74 multipliée par la durée de l'exécution séquentielle.

Pire cas

Le pire des cas correspond à une famille d'arbres non binaires où presque tous les enfants sont issus de branches gauches : il n'y a pas de branches gauches parallélisables (figure 5.9). Ce sont des ensembles de solutions du programme linéaire P_1 où de nombreuses solutions sont alignées verticalement et où nous cherchons un seul ensemble de Pareto. Presque tous les appels vers le haut de la procédure LocalPareto ne sont pas lancés.

Comme aucune branche gauche n'est parallélisable, le coefficient d'accélération est dans ce cas égal à 0.

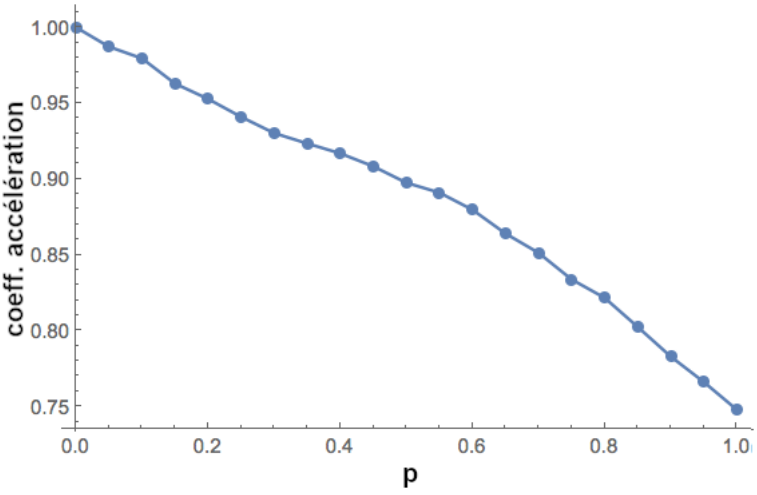

Figure 5.8 : Évolution du coefficient d'accélération d'un arbre d'exécution en fonction de la probabilité p. La génération de l'arbre est arrêtée au bout de 100 itérations. La probabilité p varie entre 0 et 1 et la probabilité q est égale à 0. Pour chaque valeur de p, nous avons généré 2000 arbres et fait la moyenne des coefficients d'accélération.

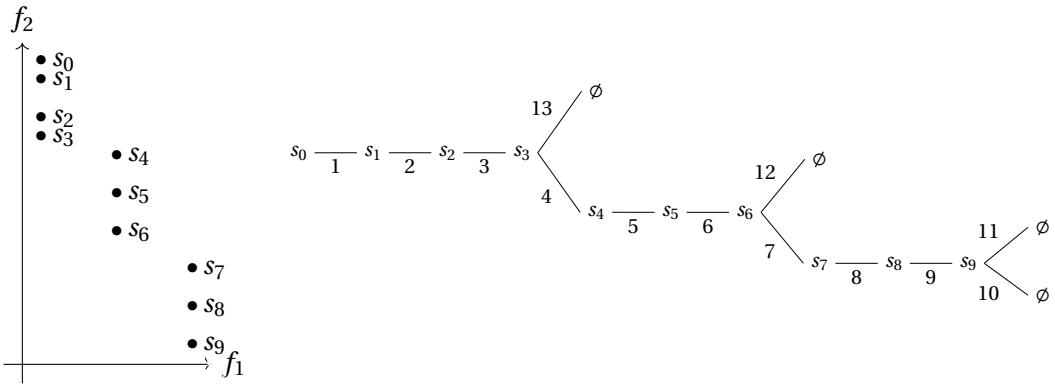

Figure 5.9 : Exemple d'exécution non parallélisable de l'algorithme FindKbestParetoSets. À gauche les solutions affichées selon leur valeur f_1 et f_2. À droite l'arbre représentant l'exécution : à chaque sommet est associé la solution trouvée durant l'appel et le poids des arêtes représente l'ordre d'exécution des appels.

5.2.5 Trouver les k meilleurs ensembles de Pareto parmi un ensemble de points

L'algorithme précédent FindKbestParetoSets (algorithme 7) peut être facilement modifié pour trouver les k meilleurs ensembles de Pareto parmi un ensemble de solutions. Le problème visant à trouver la première courbe de Pareto parmi un ensemble de points est appelé *maxima of a set of vectors* et de nombreux algorithmes existent pour le résoudre [81]. Un des premiers algorithmes proposés [119] permet de résoudre ce problème pour n critères. Cet algorithme a une complexité en temps de $\mathcal{O}(n \log_2 n)$ pour des problèmes bi ou tri-objectifs, avec n le nombre de solutions à trier.

Nous montrons ici que cet algorithme peut être adapté pour trouver les k meilleurs ensembles de Pareto pour des problèmes bi-objectifs, avec la même complexité. L'algorithme FindKbestParetoSetsScatterPlot (algorithme 8) trouve les k meilleurs ensembles de Pareto d'un ensemble de solutions, noté $\mathscr{S} = \{s_1, \ldots, s_n\}$, ayant 2 critères. Nous supposons que les deux objectifs doivent être minimisés.

Soit $\mathscr{R}$ l'ensemble des ensembles de Pareto des solutions de $\mathscr{S}$. On note $\mathscr{R}_i$ l'ensemble des solutions $s \in \mathscr{S}$ appartenant à l'ensemble de Pareto i. Le j^e élément de $\mathscr{R}_i$ est noté $\mathscr{R}_{i,j}$. La notation $s \prec \mathscr{R}_i$ signifie qu'il existe au moins une solution $s' \in \mathscr{R}_i$ telle que $f_2(s') \leq f_2(s)$. La procédure $minima$ signifie que la solution ayant la valeur minimum est retournée. En cas d'égalité, c'est la nouvelle solution qui est retournée.

Données : Un ensemble de points $\mathscr{S}$.
Résultat : Les k meilleurs ensembles de Pareto parmi $\mathscr{S}$.
1 Ordonner les éléments de $\mathscr{S}$, $s_1, \ldots, s_n$ tel que $f_1(s_1) \leq f_1(s_2) \leq \cdots \leq f_1(s_n)$. En cas d'égalité, les éléments sont ordonnés dans l'ordre croissant selon le deuxième critère ;
2 $j := 1$, $\mathscr{R}_{1,0} := \mathscr{R}_{2,0} := \cdots := \mathscr{R}_{k,0} := \emptyset$, $cp := false$;
3 **pour** i *de* 1 *à* k **faire**
4 **si** cp *OU* $s_i \prec \mathscr{R}_i$ **alors**
5 $\mathscr{R}_{i,j} := \mathscr{R}_{i,j-1}$;
6 **sinon**
7 $\mathscr{R}_{i,j} := minima(\mathscr{R}_i \cup f_2(s_j), cp := true)$;
8 **fin**
9 **fin**
10 **si** $j = n$ **alors**
11 **retourner** $\mathscr{R}$ *et STOP.*
12 **sinon**
13 $j := j + 1$, $cp := false$ et retourner à l'étape 3 ;
14 **fin**

Algorithme 8 : FindKbestParetoSetsScatterPlot

Théorème 5.2.1. *La solution s_j appartient au i^e ensemble de Pareto de $\mathscr{S}$ si et seulement si $s \in \mathscr{R}_i$.*

Démonstration. Nous savons que pour toute solution s' appartenant à l'ensemble de Pareto i, $\mathscr{R}_i$, $f_1(s') \leq f_1(s_j)$ (ligne 1).

— Si la condition $s_j \not\prec \mathscr{R}_i$ est réalisée, c'est que les solutions $s' \in \mathscr{R}_i$ sont telles que $f_2(s_j) < f_2(s')$. Donc s_j est une solution non comparable aux solutions s' et fait donc partie de $\mathscr{R}_i$.
— Si la condition $s_j \prec \mathscr{R}_i$ est réalisée, c'est qu'il existe des solutions $s' \in \mathscr{R}_i$ telles que $f_2(s_j) \geq f_2(s')$. Donc s_j est une solution dominée par des solutions $s' \in \mathscr{R}_i$ et ne peut donc faire partie de $\mathscr{R}_i$.
— Si cp est réalisée, c'est que s_j appartient à $\mathscr{R}_i$. Elle domine donc les solutions $s' \in \mathscr{R}_{>i}$.

□

La complexité de l'algorithme FindKbestParetoSetsScatterPlot est estimée à $\mathcal{O}(n\log_2 n)$ pour deux objectifs. L'étape 1 requière $\mathcal{O}(n\log_2 n)$ comparaisons pour ordonner les solutions de $\mathscr{S}$, où n est le nombre de solutions dans $\mathscr{S}$. L'étape 3 consiste à tester si $f_2(s_j) \prec \mathscr{R}_i$ jusqu'à k fois. Ce test est la simple comparaison entre $f_2(s_j)$ et l'élément $\mathscr{R}_{i,j-1}$. Enfin, le nombre de comparaisons requis est $\mathcal{O}(n\log_2 n) + \mathcal{O}(nk)$, i.e., $\mathcal{O}(nk\log_2 n)$.

5.2.6 Algorithme approché pour trouver les k meilleurs ensembles de Pareto

La complexité en temps de résolution des PLNE est exponentielle, c'est pourquoi il peut être intéressant de considérer des algorithmes approchés. Dans cette section, nous supposons qu'une solution $(1+\alpha)$-approchée peut être trouvée au programme linéaire résolu lors de la procédure LocalPareto (procédure 2). Nous montrons qu'une variante de notre algorithme FindKbestParetoSets (algorithme 7) peut retourner des ensembles de Pareto $(1+\alpha)^k$-approchés (voir définition 1.2.17, page 1.2.17). Pour cela, nous montrons tout d'abord qu'une variante de la méthode *ϵ-constraint*, sur lequel est basé notre algorithme FindKbestParetoSets, peut retourner un front de Pareto $(1+\alpha)$-approché.

Méthode *ϵ-constraint* $(1+\alpha)$-approchée

Nous proposons une variante de la méthode *ϵ-constraint* (voir algorithme 9) pour résoudre un programme linéaire en nombres entiers bi-objectif. Le programme linéaire est le suivant :

$$\min f_1(x)$$

$$\min f_2(x)$$

tel que :

$$g_j(x) \leq 0 \qquad j = 1,\ldots,m$$

$$x = (x_1, x_2, \ldots, x_n)$$

$$x_i \in \mathbb{Z}.$$

Dans cette variante, à chaque itération de l'algorithme, les problèmes $P_1(\epsilon_1)$ et $P_2(\epsilon_2)$ vont être résolus (lignes 5 et 7), avec ϵ_1 et ϵ_2 des constantes. Le programme linéaire $P_1(\epsilon_1)$ est le suivant :

$$\min f_1(x)$$

tel que :

$$f_2(x) \leq \epsilon_1$$

$$g_j(x) \leq 0 \qquad j = 1,\ldots,m$$

$$x = (x_1, x_2, \ldots, x_n)$$

$$x_i \in \mathbb{Z}.$$

Le programme linéaire $P_2(\epsilon_2)$ est le suivant :

$$\min f_2(x)$$

tel que :

$$f_1(x) \leq \epsilon_2$$

$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$

$$x = (x_1, x_2, \ldots, x_n)$$

$$x_i \in \mathbb{Z}.$$

Nous supposons que chacun de ces problèmes peut être $(1+\alpha)$-approché. Parmi les deux solutions trouvées, celle dominant l'autre sera utilisée dans la suite de l'algorithme. Si les solutions ne sont pas comparables, la solution ayant la valeur de f_2 maximale sera utilisée (la seconde solution sera trouvée lors d'une prochaine itération).

Cet algorithme permet d'obtenir un front de Pareto $(1+\alpha)$-approché.

Démonstration. Par définition, la première solution de F est $(1+\alpha)$-approchée. Pour chaque itération, la solution suivante est trouvée en résolvant un programme linéaire $(1+\alpha)$-approché, minimisant f_1 dans la zone en dessous de ϵ. Donc cette dernière solution est $(1+\alpha)$-approchée pour cette zone du front de Pareto. Puisque l'algorithme trouve des solutions couvrant la zone entière du front de Pareto exact, le front de Pareto trouvé par cet algorithme est $(1+\alpha)$-approché. □

La figure 5.10 illustre le principe de l'algorithme 9 pour trouver des solutions approchées.

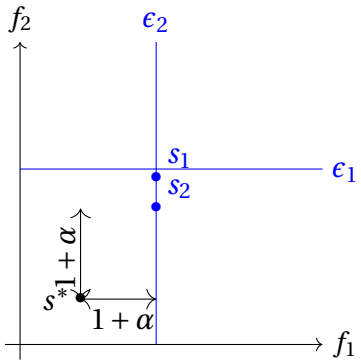

Figure 5.10 : Illustration du principe de l'algorithme 9 pour trouver des solutions aux programmes P_1 et P_2. Avec s^* une solution de la courbe de Pareto optimale.

Résultat : Un front de Pareto $(1+\alpha)$-approché.
Calculer le point idéal, noté I, et le point nadir, noté N, $(1+\alpha)$-approchés ;
$F := \{(f_1(I), f_2(N))\}$;
$\epsilon_1 = f_2(N) - \delta$ $(\delta > 0)$;
tant que $\epsilon_1 > f_2(I)$ **faire**
    $s_1 := résoudre(P_1(\epsilon_1))$; // $(1+\alpha)$-approché
    $\epsilon_2 = f_1(s_1)$;
    $s_2 := résoudre(P_2(\epsilon_2))$; // $(1+\alpha)$-approché
    si $s_1 \succ s_2$ **alors**
        $F := F \cup \{s_1\}$;
        $\epsilon_1 = f_2(s_1) - \delta$;
    sinon si $s_2 \succ s_1$ **alors**
        $F := F \cup \{s_2\}$;
        $\epsilon_1 = f_2(s_2) - \delta$;
    sinon
        si $f_2(s_1) \geq f_2(s_2)$ **alors**
            $F := F \cup \{s_1\}$;
            $\epsilon_1 = f_2(s_1) - \delta$;
        sinon
            $F := F \cup \{s_2\}$;
            $\epsilon_= f_2(s_2) - \delta$;
        fin
    fin
fin
Supprimer les solutions dominées de F ;

Algorithme 9 : Méthode ϵ-*constraint* bi-objectif $(1+\alpha)$-approchée.

Algorithme $(1+\alpha)$-approché pour trouver les k meilleurs ensembles de Pareto

De la même manière que précédemment, nous proposons la variante suivante de notre algorithme pour trouver les k meilleurs ensembles de Pareto (algorithme 10). Dans cette variante, à chaque appel de la procédure Local-Pareto, les problèmes $P_1(\lambda^-, \lambda^+, F)$ et $P_2(\lambda^-, \lambda^+, F)$ sont résolus de manière $(1+\alpha)$-approchée.

Le programme linéaire $P_1(\lambda^-, \lambda^+, F)$ est le suivant :

$$\min f_1(x)$$

tel que :

$$f_2(x) \geq \lambda^-$$

$$f_2(x) \leq \lambda^+$$

$$\text{DIFF}(s) \forall s \in F$$

$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$

$$x = (x_1, x_2, \ldots, x_n)$$

$$x_i \in \mathbb{Z}.$$

Le programme linéaire $P_2(\lambda^-, \lambda^+, F)$ est le suivant :

$$\min f_2(x)$$

tel que :

$$f_2(x) \geq \lambda^-$$

$$f_2(x) \leq \lambda^+$$

$$\text{DIFF}(s) \forall s \in F$$

$$g_j(x) \leq 0 \qquad j = 1, \ldots, m$$

$$x = (x_1, x_2, \ldots, x_n)$$

$$x_i \in \mathbb{Z}.$$

Comme précédemment, nous supposons que chacun de ces problèmes peut être $(1+\alpha)$ approché.

Parmi les deux solutions trouvées par ces problèmes, celle dominant l'autre sera utilisée dans la suite de l'algorithme. Si les deux solutions ne sont pas comparables, celle ayant la valeur f_2 maximale sera utilisée (la seconde solution sera trouvée lors d'une prochaine itération).

Résultat : Les k meilleurs ensembles de Pareto $(1+\alpha)^k$-approchés.
1 $(1+\alpha)$-approché LocalPareto$(-\infty, +\infty, \emptyset)$; `// procédure 3`
2 $\mathscr{R} := \mathscr{R} \backslash \{x \in \mathscr{R}, l(x) > k\}$;
3 **retourner** $\mathscr{R}$

Algorithme 10 : $(1+\alpha)$-approché FindKbestParetoSets

Proposition 2. *L'algorithme 10 permet de trouver les k meilleurs ensembles de Pareto* $(1+\alpha)^k$*-approchés.*

Démonstration. La première solution trouvée entre $-\infty$ et $+\infty$ est $(1+\alpha)$-approchée. La preuve pour les appels vers le bas (ligne 25 de la procédure $(1+\alpha)$-approché LocalPareto) est identique à la preuve 5.2.6 : quand un appel vers le bas est effectué, la nouvelle solution trouvée appartient au même ensemble de Pareto que l'ancienne solution ou à un meilleur ensemble de Pareto. La garantie de performance est alors inchangée pour l'ensemble de Pareto concerné.

Lors d'un appel vers le haut, si la nouvelle solution trouvée appartient à un ensemble de Pareto d'un rang plus élevé, la garantie de performance de l'ensemble de Pareto de l'ancienne solution est multipliée par $(1+\alpha)$. Par contre, si la nouvelle solution appartient au même ensemble de Pareto, la garantie de performance est inchangée.

Puisque l'algorithme permet de trouver des solutions couvrant la zone des k meilleurs ensembles de Pareto exacts, les k meilleurs ensembles de Pareto trouvés par cet algorithme sont $(1+\alpha)^k$-approchés. □

Résultat : Une solution $(1+\alpha)$-approchée de $P_1(\lambda^-, \lambda^+, F)$.

$s_1 := résoudre(P_1(\lambda^-, \lambda^+, F))$; // $(1+\alpha)$-approché
$s_2 := résoudre(P_2(\lambda^-, \lambda^+, F))$; // $(1+\alpha)$-approché
si $s_1 \succ s_2$ **alors**
  $s := s_1$;
sinon si $s_2 \succ s_1$ **alors**
  $s := s_2$;
sinon
  si $f_2(s_1) > f_2(s_2)$ **alors**
    $s := s_1$;
  sinon
    $s := s_2$;
  fin
fin
$l(s) := 1$;
si $\mathscr{L} := \{x \in \mathscr{R}, s \prec x\} \neq \emptyset$ **alors**
  $l(s) := \max_{x \in \mathscr{L}} l(x) + 1$;
fin
si $l(s) \leq k+1$ **alors**
  $\mathscr{R} := \mathscr{R} \cup \{s\}$;
  si $(\exists x \in \mathscr{R}$ *t.q.* $f_1(x) = f_1(s)$ *ET* $x \neq s)$ *ET* $(\nexists x \in \mathscr{R}$ *t.q.* $f_1(x) = f_1(s)$ *ET* $f_2(x) = f_2(s)$ *ET* $x \neq s)$ **alors**
    pour $x \in \mathscr{R}$ *t.q.* $f_1(x) = f_1(s)$ *ET* $x \prec s$ **faire**
      $l(x) := l(x) + 1$;
    fin
  fin
  $(1+\alpha)$-approché LocalPareto$(\lambda^-, f_2(s) - \epsilon, F)$;
  si $l(s) \leq k$ **alors**
    si $\lambda^- = f_2(s)$ *OU* $F = \emptyset$ **alors**
      $(1+\alpha)$-approché LocalPareto$(f_2(s), \lambda^+, F \cup \{s\})$;
    sinon
      $(1+\alpha)$-approché LocalPareto$(f_2(s), \lambda^+, F \cup \{s\} \setminus \{dernier(F)\})$;
    fin
  fin
fin

Procédure 3 : $1+\alpha$-approché LocalPareto(λ^-, λ^+, F)

5.3 Discussion

Nous avons développé un algorithme original, appelé FindKbestParetoSets (algorithme 7), pour trouver les k meilleurs ensembles de Pareto exacts d'un programme linéaire bi-objectif. Notre algorithme permet de générer des solutions sous-optimales ordonnées entre elles par les relations de dominance. Il n'existe pas, à notre connaissance, d'autres méthodes permettant de trouver les k meilleurs ensembles de Pareto exacts.

Notre algorithme est générique, c'est à dire qu'il fonctionne pour tout programme linéaire bi-objectif. Nous l'avons notamment implémenté avec le programme bi-objectif de prédiction de structures secondaires d'ARN, décrit

en section 2.2.1, avec l'outil BiokoP. L'algorithme FindKbestParetoSets décrit dans ce chapitre, présente quelques améliorations qui permettent de diminuer la complexité de l'algorithme par rapport à celui implémenté dans BiokoP. Néanmoins, l'implémentation de l'algorithme dans BiokoP nous a permis de tester cet algorithme en condition réelle. Le programme linéaire bi-objectif résolut dans BiokoP a une complexité importante, et nous avons donc pu constater que le temps d'exécution de notre algorithme pouvait être long. Nous avons alors exploré différentes pistes.

Nous nous sommes tout d'abord intéressés à la parallélisation de l'algorithme FindKbestParetoSets. Certaines parties de l'algorithme peuvent être parallélisées : certaines résolutions du programme linéaire peuvent être effectuées en même temps. Les autres résolutions du programme linéaire ne peuvent pas être parallélisée car elles dépendent de résolutions précédentes pour pouvoir être lancées. Cette parallélisation de l'algorithme permet d'accélérer le temps d'exécution avec un facteur de 3/4.

Nous nous sommes ensuite intéressés aux algorithmes approchés. En supposant que le programme linéaire pouvait être résolut de manière $(1+\alpha)$-approchée, l'algorithme que nous proposons permet de trouver les k meilleurs ensembles de Pareto $(1+\alpha)^k$-approchés. Cela nous a également permis de montrer que la méthode ϵ-*constraint* pouvait être $(1+\alpha)$-approchée, avec la même supposition.

Nous nous sommes également penchés sur un problème connexe à celui de trouver les k meilleurs ensembles de Pareto exacts d'un programme linéaire bi-objectif : trouver les k meilleurs ensembles de Pareto exacts parmi un ensemble de points. Nous disposons d'un ensemble de solutions d'un problème bi-objectif et nous cherchons à déterminer à quel ensemble de Pareto chacune appartient. Pour cela, nous avons proposé un algorithme appelé FindKbestParetoSetsScatterPlot.

Conclusion et perspectives

Conclusion

Ces trois années de thèse ont permis de développer des méthodes et des outils originaux dédiés à la prédiction de structures secondaires d'ARN et de complexes d'ARN. Nous nous sommes plus particulièrement intéressés à la prédiction de structures à partir de séquences d'ARN seuls. Nous avons pu développer trois méthodes et outils de bioinformatique, à savoir :

- une méthode pour la prédiction de structures secondaires d'un ARN : l'outil BiokoP ;
- une méthode pour la prédiction de structures secondaires de complexes d'ARN : l'outil RCPred ;
- une méthode pour la prédiction de structures secondaires de complexes d'ARN, intégrant des données structurales comme le SHAPE, DMS, ou le PARS, et intégrant des contraintes utilisateurs : l'outil C-RCPred.

Ces méthodes sont basées sur l'approche thermodynamique permettant de prédire la structure secondaire d'un ARN à partir de sa séquence.

Nos méthodes sont capables de prédire des structures secondaires contenant des motifs particuliers : les pseudonœuds. Ces motifs sont parfois considérés comme faisant partie de la structure tertiaire et sont le plus souvent ignorés pour la prédiction de structures secondaires d'ARN. Ces motifs sont en effet plus difficiles à prédire car les paramètres énergétiques du modèle thermodynamique ne sont pas assez précis et la complexité des algorithmes basés sur la programmation dynamique est élevée. À cause de cela, la plupart des outils utilisent des simplifications du modèle thermodynamique pour les prédire.

Nos méthodes permettent également de prédire des structures sous-optimales qui sont très importantes dans la prédiction de structures d'ARN. En effet, les ARN sont dynamiques, ils peuvent adopter plusieurs structures suivant leur fonction biologique, leurs interactions ou plus simplement leur environnement dans la cellule. Ainsi, il est important de prédire un ensemble de structures et non pas l'unique structure d'énergie libre minimum. De plus, même lorsqu'un ARN ne peut adopter qu'une seule structure, elle ne correspond pas toujours à la structure d'énergie libre minimum mais à une structure proche de celle-ci et est donc sous-optimale.

Une originalité de notre approche est de combiner différents modèles de prédiction de structures secondaires de la littérature. Ainsi BiokoP combine les deux modèles MFE (*Minimum Free Energy*) et MEA (*Maximum Expected Accuracy*). Nous tirons profit de chacun des modèles pour trouver les bonnes structures d'un ensemble d'ARN. En effet, BiokoP peut prédire toutes les structures des deux modèles séparément et il peut également prédire de nouvelles structures grâce à la combinaison des deux modèles.

La problématique bioinformatique de l'outil BiokoP nécessitait de résoudre un programme linéaire en nombres entiers (PLNE) bi-objectif, en trouvant des solutions sous-optimales. Cela nous a conduit à développer un nouvel algorithme générique pour trouver les k meilleurs ensembles de Pareto exacts d'un PLNE bi-objectif. Comme résoudre un PLNE est exponentiel, nous avons développé une version parallélisée de l'algorithme. Nous avons étudié de manière théorique le gain moyen concernant le temps d'exécution de la version parallèle par rapport à la version séquentielle. Nous avons, de plus, développé une version approchée avec garantie de performance de cet algorithme ainsi que de la méthode *ϵ-constraint*, sur laquelle est basée notre algorithme. Par ailleurs, nous avons développé un algorithme exact pour résoudre un problème connexe (qui est une généralisation d'un problème précédemment connu dans la littérature) : trouver les k meilleurs ensembles de Pareto parmi un ensemble de points.

Nous avons testé notre méthode avec l'outil BiokoP sur 198 ARN ayant des pseudonœuds de tous les types, et sur 145 ARN sans pseudonœuds. Les structures secondaires de ces ARN sont validées expérimentalement. Les tests nous ont permis de montrer que BiokoP est capable de prédire des structures plus proches de la structure de référence que les outils de la littérature pour les ARN avec pseudonœuds. Nous avons également montré qu'il prédit de manière homogène tous les types de pseudonœuds, avec un F_1-score moyen d'environ 70%. BiokoP est disponible sur la plateforme EvryRNA en tant que webserver.

RCPred et C-RCPred prédisent des structures secondaires de complexes d'ARN à partir d'un ensemble de structures secondaires prédéfinies pour chaque ARN du complexe et d'un ensemble de sites d'interactions par paire d'ARN. Ces ensembles sont représentés par un graphe où il faut trouver la clique maximum, correspondant à une structure d'un complexe. Ces outils retournent les cliques ayant les meilleurs poids, c'est-à-dire, ayant les plus basses énergies libres. L'outil C-RCPred retourne les cliques également suivant des données structurales et des contraintes utilisateurs.

Le problème de la clique étant NP-difficile, nous nous sommes basés sur une heuristique performante et récente appelée *Breakout Local Search* (BLS), elle-même basée sur la recherche locale et la recherche tabou. Nous l'avons adaptée en deux étape. Tout d'abord, pour RCPred, il fallait adapter l'heuristique pour générer plusieurs solutions. Ensuite, pour C-RCPred, il fallait rendre l'heuristique multi-critère afin de retourner un ensemble de Pareto approché.

Nous avons testé RCPred sur 90 complexes d'ARN dont les structures secondaires sont validées expérimentalement. Les tests réalisés montrent que RCPred prédit les structures de complexes d'ARN plus précisément que les outils de la littérature, avec un MCC moyen d'environ 60%. Ces résultats montrent que la

modélisation du problème de prédiction de structures secondaires sous forme d'un problème de graphe est intéressante et permet de tirer parti des nombreux outils de prédiction de structures secondaires d'ARN et d'interactions entre deux ARN.

RCPred est disponible sur la plateforme EvryRNA en tant que webserver. Un ensemble d'outils de prédiction de structures secondaires et de sites d'interaction sont disponibles pour générer les entrées de RCPred. L'utilisateur peut également choisir de n'utiliser aucun outil et de fournir ses propres structures et interactions. Les structures secondaires de complexes prédites par RCPred sont affichées dynamiquement avec l'outil forna.

La méthode développée avec l'outil C-RCPred a permis d'intégrer des données structurales et des contraintes utilisateurs à une méthode de prédiction de structures de complexes d'ARN. Pour intégrer ces données structurales, nous nous sommes basés sur la définition d'un MCC pour maximiser l'accord entre les données et les structures de complexes prédites. Les contraintes utilisateurs respectées par les structures prédites sont calculées grâce à un indice de confiance donné pour chaque contrainte et à partir desquels nous calculons une proportion de contraintes respectées par structure prédite. Grâce à ces deux nouveaux critères, nous obtenons un problème tri-critère pour lequel nous avons développé une heuristique basée sur la BLS.

D'un point de vue algorithmique nous nous sommes intéressés à un second problème de graphe, la détermination du nombre chromatique, pour limiter la profondeur des pseudonœuds des complexes dans C-RCPred. En effet les structures secondaires d'ARN peuvent être modélisée par des graphes de cercle. Le nombre chromatique d'un de ces graphes donne la profondeur maximum des pseudonœuds formés par les hélices.

C'est également à cette occasion que nous avons développé une méthode pour classer les structures de complexes d'ARN car la génération du front de Pareto apporte un nombre de structures pouvant être important. Pour cela nous nous sommes intéressés aux méthodes de classification à noyau et avons utilisé le noyau NSPDK qui a été développé pour calculer les distances entre graphes représentant des molécules chimiques. Ce noyau a été utilisé plusieurs fois par des outils de classification de structures secondaires d'ARN. Nous avons utilisé la méthode des k-moyennes avec ce noyau (classification structurale) pour classer les structures générées par notre heuristique.

Les résultats obtenus avec l'outil C-RCPred sont similaires à ceux de RCPred. Ils nous ont permis de montrer que notre modélisation des contraintes utilisateurs en tant qu'objectif est pertinente et également que notre méthode de classification, ainsi que le contrôle du niveau des pseudonœuds sont efficaces. Cependant, en l'état actuel, nous avons montré que le fait d'autoriser des conflits entre les appariements d'un complexe n'était pas concluant. Enfin, nous n'avons pas pu tester réellement l'ajout de données structurales car nous disposions de trop peu de données et celles que nous avions ne correspondaient pas aux structures de références des complexes que nous avions récupérés.

Le développement d'un webserver pour C-RCPred est en cours de finalisation, il sera très prochainement disponible sur la plateforme EvryRNA. Le

webserver est basé sur celui de RCPred et permet en plus d'ajouter des contraintes et des données structurales. Pour C-RCPred, davantage d'outils seront disponibles pour générer les structures secondaires et interactions d'entrée, et notamment des outils prenant en compte des données structurales. De plus, la visualisation des structures de complexes prédites est changée. Elle est maintenant plus dynamique grâce à l'outil Ribosketch qui gère mieux l'affichage des structures de complexes. Nous avons modifié également Ribosketch afin de pouvoir ajouter des contraintes utilisateurs dynamiquement. L'utilisateur peut ainsi ajouter ou supprimer des appariements, ou des motifs pour contraindre les futures prédictions de C-RCPred.

Perspectives

Cette thèse ouvre de nombreuses perspectives, que ce soit pour chaque méthode développée ou plus globalement pour la problématique de prédiction de structures d'ARN ou de complexes d'ARN.

Une limite de l'approche utilisée pour l'outil BiokoP est la complexité en temps de l'algorithme. Nous avons montré qu'il était possible de paralléliser l'algorithme, ceci pourrait être utilisé pour accélérer le temps d'exécution du programme. Cependant, le temps de résolution exact du PLNE serait toujours exponentiel, et même si celui-ci peut être parallélisé avec des solveurs comme CPLEX, il demeure important pour des séquences d'ARN ayant une longueur supérieure à 150 nucléotides.

Une autre possibilité est de relaxer ce PLNE. En effet, les programmes linéaire à variables continues sont résolus très rapidement. Nous nous sommes intéressés notamment à une méthode de relaxation appelée la relaxation Lagrangienne [15]. Cependant, ces relaxations n'ont pas permis de déboucher sur des prédictions de structures réalistes en relachant trop les contraintes imposées.

Une autre idée pour diminuer la complexité de résolution du PLNE est de lui ajouter des contraintes pour réduire grandement l'espace des solutions (par exemple si une partie de la structure est connue ou si on connait déjà l'existence d'un pseudonœud ou son type). La difficulté consiste à pouvoir écrire sous forme de contraintes linéaires ce type de contraintes sur l'existence de pseudonœuds, leur type, ou encore l'existence d'autres motifs.

Dans l'outil BiokoP nous combinons deux modèles MFE et MEA. Une perspective est de combiner plus de deux modèles de prédiction pour améliorer la qualité des prédictions. Nous pourrions également combiner d'autres critères comme nous l'avons fait pour l'outil C-RCPred. Pour combiner plus de deux modèles, l'algorithme FindKbestParetoSets devrait être modifié pour pouvoir accepter $n > 2$ critères. Cela augmenterait la complexité du programme linéaire à résoudre, nécessitant ainsi l'utilisation de techniques de relaxation et/ou de parallélisation évoquées précédemment.

Enfin, une dernière perspective pour l'outil BiokoP serait d'intégrer les données structurales de SHAPE, DMS ou PARS, ainsi que des contraintes utilisateurs, comme cela a été fait pour l'outil C-RCPred. La classification des

structures prédites par BiokoP pourrait également être étudiée car les structures prédites peuvent être similaires.

Une perspective est de proposer aux utilisateurs de notre plateforme web, EvryRNA, un outil indépendant permettant de classer les structures secondaires de complexes d'ARN. En effet il n'existe pas d'outil de ce type à notre connaissance. Cet outil serait basé sur la méthode que nous avons développée avec C-RCPred, c'est-à-dire sur la classification spectrale et le noyau NSPDK.

Une perspective pour l'outil RCPred est d'utiliser l'optimisation multi-objectif pour combiner les modèles de prédiction, comme effectué dans BiokoP. Nous pensons plus particulièrement au modèle MEA, comme effectué dans l'outil BiokoP. En effet, nous avions montré que combiner ces deux modèles MFE et MEA, permettait de trouver un plus large spectre de structures secondaires et notamment de nouvelles structures qui ne peuvent être trouvées avec chaque modèle indépendamment.

Un des critères actuels de C-RCPred est l'accord avec des données biologiques structurales. Comme nous l'avons évoqué précédemment, nous n'avons pas pu effectuer de tests concluants. Nous aimerions donc réaliser des tests supplémentaires avec de nouvelles données structurales et de nouveaux complexes, ainsi qu'avec des données artificielles, pour évaluer notre prise en compte de données structurales.

Par ailleurs, nous aimerions réaliser une étude pour déterminer quel est le niveau d'approximation moyen de l'ensemble de Pareto généré par C-RCPred. En effet, pour RCPred, nous avons montré que la solution retournée par l'heuristique correspondait dans la plupart des cas à la solution optimale du problème de clique. Si l'heuristique de C-RCPred permet de trouver un ensemble de Pareto proche de l'ensemble de Pareto exact, nous pourrions générer des ensembles de Pareto sous-optimaux approchés grâce à cette heuristique.

Pour les outils RCPred et C-RCPred, une perspective est d'améliorer le calcul de l'énergie libre. En effet, le calcul de l'énergie libre d'un complexe est effectué en calculant la somme des énergies libres des structures et sites d'interactions le composant. Cela apporte un biais ; en effet, ce calcul est une estimation de l'énergie libre. Le calcul de l'énergie libre de l'outil NUPACK est effectué de manière précise. Il peut être effectué pour un complexe d'ARN mais sans pseudonœuds. Nous aimerions ajuster le calcul de l'énergie de nos deux outils pour améliorer la qualité des résultats. Une possibilité serait de calculer l'énergie résultante de deux interactions et/ou structures et d'en tenir compte en mettant un poids sur chaque arête du graphe. Nous aurions alors un problème de clique multi-objectif avec des poids sur les sommets et les arêtes. Le calcul de l'énergie générerait toujours un biais mais celui-ci serait moindre.

Nous aimerions également améliorer la modélisation de l'autorisation de conflits dans les structures secondaires de complexes. En effet, les tests effectués avec C-RCPred nous ont montré que quelques conflits autorisés ne permettaient pas d'améliorer les résultats. Nous pensons que cela vient de la différence d'énergie lorsque des conflits entre structures/interactions font supprimer des appariements. L'idée est donc également de mettre l'énergie libre, ou une différence d'énergie libre, sur les arêtes du graphe.

Une autre perspective serait de prédire des structures de complexes ARN-

protéines. En effet, il existe de nombreux complexes de ce type dans les cellules. Notre modélisation par un problème de graphe permettrait d'adapter facilement notre méthode à ce type de complexe.

Enfin, intégrer des motifs 3D aux structures prédites par RCPred et C-RCPred pourrait être intéressant. En effet, des bases de données de motifs 3D existent, comme CaRNAval [181], RNA 3D ATLAS [165] ou RNAmotif [64], et il a été montré qu'insérer des motifs dans le mécanisme de prédiction pouvait améliorer les prédictions de structures secondaires [180]. Cela permettrait d'améliorer la modélisation 3D des structures secondaires de complexes prédites.

Une dernière perspective, plus globale, serait d'intégrer des données de co-expression. Il existe de telles données pour les ARN et les protéines. Si un ensemble d'ARN et/ou de protéines ne sont pas exprimés en même temps dans la cellule, ils ne peuvent pas former de complexe. Si nous avons des données de co-expression, nous pourrions à partir de ces données identifier des groupes d'ARN et de protéines exprimés conjointement. Ensuite, pour chaque groupe, nous pourrions, avec notre méthode de prédiction de structure de complexe, essayer de déterminer quelles structures seraient susceptibles de se former dans la cellule.

Bibliographie

[1] Gnu linear programming kit, version x.y. `http://www.gnu.org/software/glpk/glpk.html`.

[2] Emile Aarts and Jan Karel Lenstra. *Local Search Algorithms*. Wiley, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 1997.

[3] Phaedra Agius, Kristin P Bennett, and Michael Zuker. Comparing RNA secondary structures using a relaxed base-pair score. *RNA*, 16(5) :865–878, 2010.

[4] Cagri Aksay, Raheleh Salari, Emre Karakoc, Can Alkan, and S Cenk Sahinalp. taveRNA : a web suite for RNA algorithms and applications. *Nucleic Acids Research*, 35(suppl_2) :W325–W329, 2007.

[5] Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1) :45–62, 2000.

[6] Can Alkan, Emre Karakoc, Joseph H Nadeau, S Cenk Sahinalp, and Kaizhong Zhang. RNA–RNA interaction prediction and antisense RNA target search. *Journal of Computational Biology*, 13(2) :267–282, 2006.

[7] Julien Allali and Marie-France Sagot. A multiple layer model to compare RNA secondary structures. *Software : Practice and Experience*, 38(8) :775–792, 2008.

[8] Rafael G Amado, Ronald T Mitsuyasu, Joseph D Rosenblatt, Frances K Ngok, Andreas Bakker, Steve Cole, Nathalie Chorn, Lii-Shin Lin, Gregory Bristol, Maureen P Boyd, et al. Anti-human immunodeficiency virus hematopoietic progenitor cell-delivered ribozyme in a phase i study : Myeloid and lymphoid reconstitution in human immunodeficiency virus type-1–infected patients. *Human gene therapy*, 15(3) :251–262, 2004.

[9] Fabian Amman, Stephan H Bernhart, Gero Doose, Ivo L Hofacker, Jing Qin, Peter F Stadler, and Sebastian Will. The trouble with long-range base pairs in RNA folding. In *Brazilian Symposium on Bioinformatics*, pages 1–11. Springer, 2013.

[10] Mirela Andronescu, Zhi Chuan Zhang, and Anne Condon. Secondary structure prediction of interacting RNA molecules. *Journal of Molecular Biology*, 345(5) :987–1001, 2005.

[11] Mirela Andronescu, Vera Bereg, Holger H Hoos, and Anne Condon. RNA STRAND : the RNA secondary structure and statistical analysis database. *BMC Bioinformatics*, 9(1) :340, 2008.

[12] Mirela S Andronescu, Cristina Pop, and Anne E Condon. Improved free energy parameters for RNA pseudoknotted secondary structure prediction. *RNA*, 16(1) :26–42, 2010.

[13] Gayatri Arun, Sarah D Diermeier, and David L Spector. Therapeutic targeting of long non-coding RNAs in cancer. *Trends in molecular medicine*, 24(3) :257–277, 2018.

[14] Egon Balas and Robert Jeroslow. Canonical cuts on the unit hypercube. *SIAM Journal on Applied Mathematics*, 23(1) :61–69, 1972.

[15] John E Beasley. Lagrangian relaxation. In *Modern heuristic techniques for combinatorial problems*, pages 243–303. John Wiley & Sons, Inc., 1993.

[16] Stanislav Bellaousov and David H Mathews. ProbKnot : fast prediction of RNA secondary structure including pseudoknots. *RNA*, 16(10) :1870–1880, 2010.

[17] Una Benlic and Jin-Kao Hao. Breakout local search for maximum clique problems. *Computers & Operations Research*, 40(1) :192–206, 2013.

[18] Helen M Berman, Wilma K Olson, David L Beveridge, John Westbrook, Anke Gelbin, Tamas Demeny, Shu-Hsin Hsieh, AR Srinivasan, and Bohdan Schneider. The nucleic acid database. a comprehensive relational database of three-dimensional structures of nucleic acids. *Biophysical journal*, 63(3) :751, 1992.

[19] Helen M Berman, Philip E Bourne, John Westbrook, and Christine Zardecki. The protein data bank. In *Protein Structure*, pages 394–410. CRC Press, 2003.

[20] Karl Bertram, Dmitry E Agafonov, Olexandr Dybkov, David Haselbach, Majety N Leelaram, Cindy L Will, Henning Urlaub, Berthold Kastner, Reinhard Lührmann, and Holger Stark. Cryo-EM structure of a pre-catalytic human spliceosome primed for activation. *Cell*, 170(4) : 701–713, 2017.

[21] Jean-François Bérubé, Michel Gendreau, and Jean-Yves Potvin. An exact ϵ-constraint method for bi-objective combinatorial optimization problems : Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research*, 194(1) :39–50, 2009.

[22] Eckart Bindewald, Tanner Kluth, and Bruce A Shapiro. Cylofold : secondary structure prediction including pseudoknots. *Nucleic Acids Research*, 38(suppl 2) :W368–W372, 2010.

[23] Eckart Bindewald, Kirill Afonin, Luc Jaeger, and Bruce A Shapiro. Multistrand RNA secondary structure prediction and nanostructure design including pseudoknots. *ACS nano*, 5(12) : 9542–9551, 2011.

[24] Eckart Bindewald, Michaela Wendeler, Michal Legiewicz, Marion K Bona, Yi Wang, Mark J Pritt, Stuart FJ Le Grice, and Bruce A Shapiro. Correlating SHAPE signatures with three-dimensional RNA structures. *RNA*, 2011.

[25] Eckart Bindewald, Kirill A Afonin, Mathias Viard, Paul Zakrevsky, Taejin Kim, and Bruce A Shapiro. Multistrand structure prediction of nucleic acid assemblies and design of RNA switches. *Nano Letters*, 16(3) :1726–1735, 2016.

[26] Ashis Kumer Biswas and Jean X Gao. PR2S2Clust : patched RNA-seq read segments' structure-oriented clustering. *Journal of bioinformatics and computational biology*, 14(05) :1650027, 2016.

[27] Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, and Helène Touzet. Alignments of RNA structures. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7 (2) :309–322, 2008.

[28] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[29] Michael Bon and Henri Orland. TT2NE : a novel algorithm to predict RNA secondary structures with pseudoknots. *Nucleic Acids Research*, 39(14) :e93–e93, 2011.

[30] Michaël Bon, Cristian Micheletti, and Henri Orland. McGenus : a Monte Carlo algorithm to predict RNA secondary structures with pseudoknots. *Nucleic Acids Research*, 41(3) :1895–1900, 2012.

[31] Anke Busch, Andreas S Richter, and Rolf Backofen. IntaRNA : efficient prediction of bacterial sRNA targets incorporating target site accessibility and seed regions. *Bioinformatics*, 24(24) : 2849–2856, 2008.

[32] Liming Cai, Russell L Malmberg, and Yunzhou Wu. Stochastic modeling of RNA pseudoknotted structures : a grammatical approach. *Bioinformatics*, 19(suppl_1) :i66–i73, 2003.

[33] Song Cao and Shi-Jie Chen. Predicting RNA pseudoknot folding thermodynamics. *Nucleic Acids Research*, 34(9) :2634–2652, 2006.

[34] Song Cao and Shi-Jie Chen. Predicting structures and stabilities for H-type pseudoknots with interhelix loops. *RNA*, 15(4) :696–706, 2009.

[35] Martin C Carlisle and Errol L Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59(3) :225–235, 1995.

[36] Robert Cedergren, Daniel Gautheret, Guy Lapalme, and François Major. A secondary and tertiary structure editor for nucleic acids. *Bioinformatics*, 4(1) :143–146, 1988.

[37] Audrey Cerqueus and Xavier Delorme. A branch-and-bound method for the bi-objective simple line assembly balancing problem. *International Journal of Production Research*, pages 1–20, 2018.

[38] Ho-Lin Chen, Anne Condon, and Hosna Jabbari. An $\mathcal{O}(n^5)$ algorithm for MFE prediction of kissing hairpins and 4-chains in nucleic acids. *Journal of Computational Biology*, 16(6) : 803–815, 2009.

[39] Xiang Chen, Si-Min He, Dongbo Bu, Fa Zhang, Zhiyong Wang, Runsheng Chen, and Wen Gao. FlexStem : improving predictions of RNA secondary structures with pseudoknots by reducing the search space. *Bioinformatics*, 24(18) :1994–2001, 2008.

[40] Shi Cheng, Yuhui Shi, and Quande Qin. On the performance metrics of multiobjective optimization. *Advances in Swarm Intelligence*, pages 504–512, 2012.

[41] Louise T Chow, James M Roberts, James B Lewis, and Thomas R Broker. A map of cytoplasmic RNA transcripts from lytic adenovirus type 2, determined by electron microscopy of RNA : DNA hybrids. *Cell*, 11(4) :819–836, 1977.

[42] Peter Clote, Stefan Dobrev, Ivan Dotu, Evangelos Kranakis, Danny Krizanc, and Jorge Urrutia. On the page number of RNA secondary structures with pseudoknots. *Journal of Mathematical Biology*, 65(6-7) :1337–1357, 2012.

[43] Pablo Cordero, Wipapat Kladwang, Christopher C VanLang, and Rhiju Das. Quantitative dimethyl sulfate mapping for automated RNA secondary structure inference. *Biochemistry*, 51(36) :7037–7039, 2012.

[44] Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*, pages 255–262. Omnipress ; Madison, WI, USA, 2010.

[45] Maria Costa and Dario Monachello. Probing RNA folding by hydroxyl radical footprinting. In *RNA Folding*, pages 119–142. Springer, 2014.

[46] Paul Dallaire and François Major. Exploring alternative RNA structure sets using MC-flashfold and db2cm. In *RNA Structure Determination*, pages 237–251. Springer, 2016.

[47] George B Dantzig and S Nash. A history of scientific computing. *Reading, Ma, USA, chapter Origins of the simplex method*, pages 141–151, 1990.

[48] Kévin Darty, Alain Denise, and Yann Ponty. VARNA : Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15) :1974, 2009.

[49] Wayne Dawson and Gota Kawai. A new entropy model for RNA : part IV, The Minimum Free Energy (MFE) and the thermodynamically most-probable folding pathway (TMPFP). *Journal of Nucleic Acids Investigation*, 5(1), 2014.

[50] Wayne Dawson, Kazuya Fujiwara, Gota Kawai, Yasuhiro Futamura, and Kenji Yamamoto. A method for finding optimal RNA secondary structures using a new entropy model (VSfold). *Nucleosides, Nucleotides, and Nucleic Acids*, 25(2) :171–189, 2006.

[51] Wayne Dawson, Kenji Yamamoto, Kentaro Shimizu, and Gota Kawai. A new entropy model for RNA : part II. Persistence-related entropic contributions to RNA secondary structure free energy calculations. *Journal of Nucleic Acids Investigation*, 4(1) :e2–e2, 2013.

[52] Wayne Dawson, Toshikuni Takai, Nobuharu Ito, Kentaro Shimizu, and Gota Kawai. A new entropy model for RNA : part III. Is the folding free energy landscape of RNA funnel shaped ? *Journal of Nucleic Acids Investigation*, 5(1), 2014.

[53] Nicolas Garreau de Loubresse, Irina Prokhorova, Wolf Holtkamp, Marina V Rodnina, Gulnara Yusupova, and Marat Yusupov. Structural basis for the inhibition of the eukaryotic ribosome. *Nature*, 513(7519) :517, 2014.

[54] Katherine E Deigan, Tian W Li, David H Mathews, and Kevin M Weeks. Accurate SHAPE-directed RNA structure prediction. *Federation of American Societies for Experimental Biology*, 23(1), 2009.

[55] Charles Delisi and Donald M Crothers. Prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences*, 68(11) :2682–2685, 1971.

[56] Clarisse Dhaenens, Julien Lemesre, and El-Ghazali Talbi. K-PPM : A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1) :45–53, 2010.

[57] Sergey M Dibrov, Jaime McLean, Jerod Parsons, and Thomas Hermann. Self-assembling RNA square. *Proceedings of the National Academy of Sciences*, 108(16) :6405–6408, 2011.

[58] Ye Ding and Charles E Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Research*, 31(24) :7280–7301, 2003.

[59] Ye Ding, Chi Yu Chan, and Charles E Lawrence. Sfold web server for statistical folding and rational design of nucleic acids. *Nucleic Acids Research*, 32(suppl_2) :W135–W141, 2004.

[60] Ye Ding, Chi Yu Chan, and Charles E Lawrence. RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, 11(8) :1157–1166, 2005.

[61] Yiliang Ding, Yin Tang, Chun Kit Kwok, Yu Zhang, Philip C Bevilacqua, and Sarah M Assmann. In vivo genome-wide profiling of RNA secondary structure reveals novel regulatory features. *Nature*, 505(7485) :696, 2014.

[62] Robert M Dirks and Niles A Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of computational chemistry*, 24(13) :1664–1677, 2003.

[63] Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM review*, 49(1) :65–88, 2007.

[64] Mahassine Djelloul. *Algorithmes de graphes pour la recherche de motifs récurrents dans les structures tertiaires d'ARN*. PhD thesis, Université Paris Sud-Paris XI, 2009.

[65] Chuong B Do, Daniel A Woods, and Serafim Batzoglou. CONTRAfold : RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14) :e90–e98, 2006.

[66] P Doty, H Boedtker, JR Fresco, R Haselkorn, and M Litt. Secondary structure in ribonucleic acids. *Proceedings of the National Academy of Sciences of the United States of America*, 45(4) : 482, 1959.

[67] Didier Dubois and Patrice Perny. A review of fuzzy sets in decision sciences : Achievements, limitations and perspectives. In *Multiple Criteria Decision Analysis*, pages 637–691. Springer, 2016.

[68] Sean R Eddy and Richard Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22(11) :2079–2088, 1994.

[69] Chantal Ehresmann, Florence Baudin, Maryléne Mougel, Pascale Romby, Jean-Pierre Ebel, and Bernard Ehresmann. Probing the structure of RNAs in solution. *Nucleic Acids Research*, 15(22) :9109–9128, 1987.

[70] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[71] Matthias Ehrgott, Thomas Stephan, and Dagmar Tenfelde-Podehl. A level set method for multiobjective combinatorial optimization : Application to the quadratic assignment problem. *Report in Wirtschaftsmathematik (WIMA Report)*, 84, 2002.

[72] Stéfan Engelen and Fariza Tahi. Tfold : efficient in silico prediction of non-coding RNA secondary structures. *Nucleic Acids Research*, 38(7) :2453–2466, 2010.

[73] Jacques R Fresco, Bruce M Alberts, Paul Doty, et al. Some molecular details of the secondary structure of ribonucleic acid. *Nature*, 188 :98–101, 1960.

[74] Eva Freyhult, Vincent Moulton, and Peter Clote. Boltzmann probability of RNA structural neighbors and riboswitch detection. *Bioinformatics*, 23(16) :2054–2062, 2007.

[75] Eva Freyhult, Vincent Moulton, and Peter Clote. RNAbor : a web server for RNA structural neighbors. *Nucleic Acids Research*, 35(suppl_2) :W305–W309, 2007.

[76] Tsukasa Fukunaga and Michiaki Hamada. RIblast : An ultrafast RNA–RNA interaction prediction system for comprehensive lncRNA interaction analysis. *BioRxiv*, page 077271, 2016.

[77] Elspeth F Garman. Developments in x-ray crystallographic structure determination of biological macromolecules. *Science*, 343(6175) :1102–1108, 2014.

[78] Christine Gaspin and Eric Westhof. An interactive framework for RNA secondary structure prediction with a dynamical treatment of constraints. *Journal of molecular biology*, 254(2) : 163–174, 1995.

[79] David P Giedroc, Carla A Theimer, and Paul L Nixon. Structure, stability and function of RNA pseudoknots involved in stimulating ribosomal frameshifting. *Journal of molecular biology*, 298(2) :167–185, 2000.

[80] Robert Giegerich, Björn Voß, and Marc Rehmsmeier. Abstract shapes of RNA. *Nucleic Acids Research*, 32(16) :4843–4851, 2004.

[81] Parke Godfrey, Ryan Shipley, and Jarek Gryz. Algorithms and analyses for maximal vector computation. *The VLDB Journal—The International Journal on Very Large Data Bases*, 16(1) : 5–28, 2007.

[82] Sam Griffiths-Jones, Simon Moxon, Mhairi Marshall, Ajay Khanna, Sean R Eddy, and Alex Bateman. Rfam : annotating non-coding RNAs in complete genomes. *Nucleic Acids Research*, 33(suppl_1) :D121–D124, 2005.

[83] Santiago Guerrero, Julien Batisse, Camille Libre, Serena Bernacchi, Roland Marquet, and Jean-Christophe Paillart. HIV-1 replication and the cellular eukaryotic translation apparatus. *Viruses*, 7(1) :199–218, 2015.

[84] Valentin Guignon, Cedric Chauve, and Sylvie Hamel. RNA StrAT : RNA Structure Analysis Toolkit. In *16th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB 2008)*, page D31, 2008.

[85] Inc. Gurobi Optimization. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2015.

[86] Christine E Hajdin, Stanislav Bellaousov, Wayne Huggins, Christopher W Leonard, David H Mathews, and Kevin M Weeks. Accurate SHAPE-directed RNA secondary structure modeling, including pseudoknots. *Proceedings of the National Academy of Sciences*, 110(14) :5498–5503, 2013.

[87] Horst W Hamacher and Maurice Queyranne. K best solutions to combinatorial optimization problems. *Annals of Operations Research*, 4(1) :123–143, 1985.

[88] Michiaki Hamada, Hisanori Kiryu, Kengo Sato, Toutai Mituyama, and Kiyoshi Asai. Prediction of RNA secondary structure using generalized centroid estimators. *Bioinformatics*, 25(4) : 465–473, 2008.

[89] Johan Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636. IEEE, 1996.

[90] Claire Herrbach. *Etude algorithmique et statistique de la comparaison des structures secondaires d'ARN*. PhD thesis, Bordeaux 1, 2007.

[91] Steffen Heyne, Fabrizio Costa, Dominic Rose, and Rolf Backofen. GraphClust : alignment-free structural clustering of local RNA secondary structures. *Bioinformatics*, 28(12) :i224–i232, 2012.

[92] Matthias Hochsmann, Thomas Toller, Robert Giegerich, and Stefan Kurtz. Local similarity in RNA secondary structures. In *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, pages 159–168. IEEE, 2003.

[93] Ivo L Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Research*, 31(13) : 3429–3431, 2003.

[94] Jiabin Huang, Rolf Backofen, and Björn Voß. Abstract folding space analysis based on helices. *RNA*, 18(12) :2135–2147, 2012.

[95] Xiaolu Huang and Hesham Ali. High sensitivity RNA pseudoknot prediction. *Nucleic Acids Research*, 35(2) :656–663, 2006.

[96] IBM. CPLEX Optimizer V12.6.3. `http://www-03.ibm.com/software/products/fr/ibmilogcpleoptistud`, n.d. Accessed : 2018-03-29.

[97] Junya Ishikawa, Hiroyuki Furuta, and Yoshiya Ikawa. RNA tectonics (tectoRNA) for RNA nanostructure design and its application in synthetic biology. *Wiley Interdisciplinary Reviews : RNA*, 4(6) :651–664, 2013.

[98] Hosna Jabbari, Ian Wark, Carlo Montemagno, and Sebastian Will. Knotty : efficient and accurate prediction of complex RNA pseudoknot structures. *Bioinformatics*, 34(22) :3849–3856, 2018.

[99] Tyler Jacks, Michael D Power, Frank R Masiarz, Paul A Luciw, Philip J Barr, and Harold E Varmus. Characterization of ribosomal frameshifting in HIV-1 gag-pol expression. *Nature*, 331(6153) :280, 1988.

[100] John A Jaeger, Douglas H Turner, and Michael Zuker. Improved predictions of secondary structures for RNA. *Proceedings of the National Academy of Sciences*, 86(20) :7706–7710, 1989.

[101] Stefan Janssen and Robert Giegerich. The RNA shapes studio. *Bioinformatics*, 31(3) :423–425, 2014.

[102] Tommy R Jensen and Bjarne Toft. *Graph coloring problems*, volume 39. John Wiley & Sons, 2011.

[103] Guosong Jiang, Ke Chen, and Jie Sun. Accurate prediction of secondary structure of tRNAs. *Biochemical and biophysical research communications*, 509(1) :64–68, 2019.

[104] Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees—an alternative to tree edit. *Theoretical Computer Science*, 143(1) :137–148, 1995.

[105] Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of computational biology*, 9(2) :371–388, 2002.

[106] Dylan F Jones, S Keyvan Mirrazavi, and Mehrdad Tamiz. Multi-objective meta-heuristics : An overview of the current state-of-the-art. *European journal of operational research*, 137(1) : 1–9, 2002.

[107] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[108] Yuki Kato, Tomoya Mori, Kengo Sato, Shingo Maegawa, Hiroshi Hosokawa, and Tatsuya Akutsu. An accessibility-incorporated method for accurate prediction of RNA–RNA interactions from sequence data. *Bioinformatics*, 33(2) :202–209, 2017.

[109] Peter Kerpedjiev, Stefan Hammer, and Ivo L Hofacker. Forna (force-directed RNA) : simple and effective online RNA secondary structure diagrams. *Bioinformatics*, 31(20) :3377–3379, 2015.

[110] Michael Kertesz, Yue Wan, Elad Mazor, John L Rinn, Robert C Nutter, Howard Y Chang, and Eran Segal. Genome-wide measurement of RNA secondary structure in yeast. *Nature*, 467 (7311) :103, 2010.

[111] Gokhan Kirlik and Serpil Sayın. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3) :479–488, 2014.

[112] Hisanori Kiryu, Taishin Kin, and Kiyoshi Asai. Rfold : an exact algorithm for computing local base pairing probabilities. *Bioinformatics*, 24(3) :367–373, 2007.

[113] Hisanori Kiryu, Goro Terai, Osamu Imamura, Hiroyuki Yoneyama, Kenji Suzuki, and Kiyoshi Asai. A detailed investigation of accessibilities around target sites of siRNAs and miRNAs. *Bioinformatics*, 27(13) :1788–1797, 2011.

[114] Dieter Klein and Edward Hannan. An algorithm for the multiple objective integer linear programming problem. *European Journal of Operational Research*, 9(4) :378–385, 1982.

[115] Gayle Knapp. Enzymatic approaches to probing of RNA secondary and tertiary structure. In *Methods in enzymology*, volume 180, pages 192–212. Elsevier, 1989.

[116] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms : A tutorial. *Reliability Engineering & System Safety*, 91(9) :992–1007, 2006.

[117] Jan Krüger and Marc Rehmsmeier. RNAhybrid : microRNA target prediction easy, fast and flexible. *Nucleic Acids Research*, 34(suppl_2) :W451–W454, 2006.

[118] Deniss Kumlander. Problems of optimization : an exact algorithm for finding a maximum clique optimized for dense graphs. In *Proceedings-Estonian Academy of Sciences Physics Mathematics*, volume 54(2), page 79. Estonian Academy Publishers, 2005.

[119] Hsiang-Tsung Kung, Fabrizio Luccio, and Franco P Preparata. On finding the maxima of a set of vectors. *Journal of the ACM (JACM)*, 22(4) :469–476, 1975.

[120] Daniel Lai and Irmtraud M Meyer. A comprehensive comparison of general RNA–RNA interaction prediction methods. *Nucleic Acids Research*, 44(7) :e61–e61, 2015.

[121] Eugene L Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management science*, 18(7) : 401–405, 1972.

[122] Audrey Legendre, Eric Angel, and Fariza Tahi. Bi-objective integer programming for RNA secondary structure prediction with pseudoknots. *BMC bioinformatics*, 19(1) :13, 2018.

[123] Audrey Legendre, Eric Angel, and Fariza Tahi. RCPred : RNA complex prediction as a constrained maximum weight clique problem. *BMC bioinformatics*, 20(3) :128, 2019.

[124] Julien Lemesre, Clarisse Dhaenens, and El-Ghazali Talbi. Parallel partitioning method (PPM) : A new exact method to solve bi-objective problems. *Computers & operations research*, 34(8) : 2450–2462, 2007.

[125] Neocles B Leontis and Eric Westhof. Geometric nomenclature and classification of RNA base pairs. *RNA*, 7(4) :499–512, 2001.

[126] Luan Lin, Wilson H McKerrow, Bryce Richards, Chukiat Phonsom, and Charles E Lawrence. Characterization and visualization of RNA secondary structure Boltzmann ensemble via information theory. *BMC bioinformatics*, 19(1) :82, 2018.

[127] Qi Liu, Victor Olman, Huiqing Liu, Xiuzi Ye, Shilun Qiu, and Ying Xu. RNACluster : An integrated tool for RNA secondary structure comparison and clustering. *Journal of computational chemistry*, 29(9) :1517–1526, 2008.

[128] Qi Liu, Yin Zhang, Ying Xu, and Xiuzi Ye. Fuzzy kernel clustering of RNA secondary structure ensemble using a novel similarity metric. *Journal of Biomolecular Structure and Dynamics*, 25(6) :685–696, 2008.

[129] Banu Lokman and Murat Köksalan. Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization*, 57(2) :347–365, 2013.

[130] Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1) :26, 2011.

[131] David Loughrey, Kyle E Watters, Alexander H Settle, and Julius B Lucks. SHAPE-Seq 2.0 : systematic optimization and extension of high-throughput chemical probing of RNA secondary structure with next generation sequencing. *Nucleic Acids Research*, 42(21) :e165–e165, 2014.

[132] Justin T Low and Kevin M Weeks. SHAPE-directed RNA secondary structure prediction. *Methods*, 52(2) :150–158, 2010.

[133] Jacob S Lu, Eckart Bindewald, Wojciech K Kasprzak, and Bruce A Shapiro. RiboSketch : versatile visualization of multi-stranded RNA and DNA secondary structure. *Bioinformatics*, 34(24) :4297–4299, 2018.

[134] Zhi John Lu, Jason W Gloor, and David H Mathews. Improved RNA secondary structure prediction by maximizing expected pair accuracy. *RNA*, 15 :1805–1813, 2009.

[135] Rune B Lyngsø and Christian NS Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of computational biology*, 7(3-4) :409–427, 2000.

[136] Nicholas R Markham and Michael Zuker. UNAFold. In *Bioinformatics*, pages 3–31. Springer, 2008.

[137] Alessio Massaro, Marcello Pelillo, and Immanuel M Bomze. A complementary pivoting approach to the maximum weight clique problem. *SIAM Journal on Optimization*, 12(4) : 928–948, 2002.

[138] David H Mathews, Mark E Burkard, Susan M Freier, Jacqueline R Wyatt, and Douglas H Turner. Predicting oligonucleotide affinity to nucleic acid targets. *RNA*, 5(11) :1458–1469, 1999.

[139] David H Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of molecular biology*, 288(5) :911–940, 1999.

[140] David H Mathews, Matthew D Disney, Jessica L Childs, Susan J Schroeder, Michael Zuker, and Douglas H Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences*, 101(19) :7287–7292, 2004.

[141] David M Mauger, Michael Golden, Daisuke Yamane, Sara Williford, Stanley M Lemon, Darren P Martin, and Kevin M Weeks. Functionally conserved architecture of hepatitis C virus RNA genomes. *Proceedings of the National Academy of Sciences*, 112(12) :3692–3697, 2015.

[142] George Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, 213(2) : 455–465, 2009.

[143] George Mavrotas and Kostas Florios. An improved version of the augmented ε-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation*, 219(18) :9652–9669, 2013.

[144] John S McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers : Original Research on Biomolecules*, 29(6-7) : 1105–1119, 1990.

[145] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4) :575–601, 1992.

[146] Dirk Metzler and Markus E Nebel. Predicting RNA secondary structures with pseudoknots by MCMC sampling. *Journal of mathematical biology*, 56(1-2) :161–181, 2008.

[147] Milad Miladi, Alexander Junge, Fabrizio Costa, Stefan E Seemann, Jakob Hull Havgaard, Jan Gorodkin, and Rolf Backofen. RNAscClust : clustering RNA sequences using structure conservation and graph based motifs. *Bioinformatics*, 33(14) :2089–2096, 2017.

[148] Milad Miladi, Soheila Montaseri, Rolf Backofen, and Martin Raden. Integration of accessibility data from structure probing into RNA-RNA interaction prediction. *bioRxiv*, page 359323, 2018.

[149] Fusako Miyamoto and Eiichi N Kodama. Novel HIV-1 fusion inhibition peptides : designing the next generation of drugs. *Antiviral Chemistry and Chemotherapy*, 22(4) :151–158, 2012.

[150] Saad Mneimneh and Syed Ali Ahmed. Gibbs/MCMC sampling for multiple RNA interaction with sub-optimal solutions. In *International Conference on Algorithms for Computational Biology*, pages 78–90. Springer, 2016.

[151] Soheila Montaseri, Fatemeh Zare-Mirakabad, and Nasrollah Moghadam-Charkari. RNA–RNA interaction prediction using genetic algorithm. *Algorithms for Molecular Biology*, 9(1) :17, 2014.

[152] Steven R Morgan and Paul G Higgs. Barrier heights between ground states in a model of RNA secondary structure. *Journal of Physics A : Mathematical and General*, 31(14) :3153, 1998.

[153] Markus E Nebel and Frank Weinberg. Algebraic and combinatorial properties of common RNA pseudoknot classes with applications. *Journal of Computational Biology*, 19(10) :1134–1150, 2012.

[154] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering : Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[155] Eugene WM Ng, David T Shima, Perry Calias, Emmett T Cunningham Jr, David R Guyer, and Anthony P Adamis. Pegaptanib, a targeted anti-VEGF aptamer for ocular vascular disease. *Nature reviews drug discovery*, 5(2) :123, 2006.

[156] Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proceedings of the National Academy of Sciences*, 77(11) :6309–6313, 1980.

[157] Ruth Nussinov, George Pieczenik, Jerrold R Griggs, and Daniel J Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied mathematics*, 35(1) :68–82, 1978.

[158] Patric RJ Östergård. A new algorithm for the maximum-weight clique problem. *Electronic Notes in Discrete Mathematics*, 3 :153–156, 1999.

[159] Aïda Ouangraoua, Pascal Ferraro, Laurent Tichit, and Serge Dulucq. Local similarity between quotiented ordered trees. *Journal of Discrete Algorithms*, 5(1) :23–35, 2007.

[160] Zhengqing Ouyang, Michael P Snyder, and Howard Y Chang. SeqFold : genome-scale reconstruction of RNA secondary structure integrating high-throughput sequencing data. *Genome Research*, 23(2) :377–387, 2013.

[161] Melih Özlen and Meral Azizoğlu. Multi-objective integer programming : a general approach for generating all non-dominated solutions. *European Journal of Operational Research*, 199 (1) :25–35, 2009.

[162] Melih Ozlen, Benjamin A Burton, and Cameron AG MacRae. Multi-objective integer programming : An improved recursive algorithm. *Journal of Optimization Theory and Applications*, 160(2) :470–482, 2014.

[163] Christos H Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 86–92. IEEE, 2000.

[164] Marc Parisien and Francois Major. The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data. *Nature*, 452(7183) :51, 2008.

[165] Anton I Petrov, Craig L Zirbel, and Neocles B Leontis. Automated classification of RNA 3D motifs and the RNA 3D motif atlas. *Rna*, 19(10) :1327–1340, 2013.

[166] Clemens Plaschka, Pei-Chun Lin, Clément Charenton, and Kiyoshi Nagai. Prespliceosome structure provides insights into spliceosome assembly and regulation. *Nature*, 559(7714) : 419, 2018.

[167] Unyanee Poolsap, Yuki Kato, and Tatsuya Akutsu. Prediction of RNA secondary structure with pseudoknots using integer programming. *BMC bioinformatics*, 10(Suppl 1) :S38, 2009.

[168] Unyanee Poolsap, Yuki Kato, and Tatsuya Akutsu. Dynamic programming algorithms for RNA structure prediction with binding sites. In *Biocomputing 2010*, pages 98–107. World Scientific, 2010.

[169] Jeffrey Ryan Proctor. *CoFold : an RNA structure prediction method that takes co-transcriptional folding into account*. PhD thesis, University of British Columbia, 2012.

[170] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3) :149–165, 2010.

[171] Joseph D Puglisi, Ruoying Tan, Barbara J Calnan, Alan D Frankel, et al. Conformation of the TAR RNA–arginine complex by NMR spectroscopy. *Science*, 257(5066) :76–80, 1992.

[172] Wayne Pullan. Approximating the maximum vertex/edge weighted clique using local search. *Journal of Heuristics*, 14(2) :117–134, 2008.

[173] Scott Quarrier, Joshua S Martin, Lauren Davis-Neulander, Arthur Beauregard, and Alain Laederach. Evaluation of the information content of RNA structure mapping data for secondary structure prediction. *RNA*, 2010.

[174] Miha Ravber, Marjan Mernik, and Matej Črepinšek. The impact of quality indicators on the rating of multi-objective evolutionary algorithms. *Applied Soft Computing*, 55 :265–275, 2017.

[175] Janina Reeder, Jens Reeder, and Robert Giegerich. Locomotif : from graphical motif description to RNA motif search. *Bioinformatics*, 23(13) :i392–i400, 2007.

[176] Jens Reeder and Robert Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC bioinformatics*, 5(1) :104, 2004.

[177] Jens Reeder and Robert Giegerich. RNA secondary structure analysis using the RNAshapes package. *Current protocols in bioinformatics*, 26(1) :12–8, 2009.

[178] Elizabeth E Regulski and Ronald R Breaker. In-line probing analysis of riboswitches. In *post-transcriptional gene regulation*, pages 53–67. Springer, 2008.

[179] Christian M Reidys, Fenix WD Huang, Jørgen E Andersen, Robert C Penner, Peter F Stadler, and Markus E Nebel. Topology and prediction of RNA pseudoknots. *Bioinformatics*, 27(8) : 1076–1085, 2011.

[180] Vladimir Reinharz, François Major, and Jérôme Waldispühl. Towards 3D structure prediction of large RNA molecules : an integer programming framework to insert local 3D motifs in RNA secondary structure. *Bioinformatics*, 28(12) :i207–i214, 2012.

[181] Vladimir Reinharz, Antoine Soulé, Eric Westhof, Jérôme Waldispühl, and Alain Denise. Mining for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network families. *Nucleic Acids Research*, 46(8) :3841–3851, 2018.

[182] Jihong Ren, Baharak Rastegari, Anne Condon, and Holger H Hoos. HotKnots : heuristic prediction of RNA secondary structures including pseudoknots. *RNA*, 11(10) :1494–1504, 2005.

[183] Jessica S Reuter and David H Mathews. RNAstructure : software for RNA secondary structure prediction and analysis. *BMC bioinformatics*, 11(1) :129, 2010.

[184] Elena Rivas and Sean R Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology*, 285(5) :2053–2068, 1999.

[185] Emily Rogers and Christine E Heitsch. Profiling small RNA reveals multimodal substructural signals in a boltzmann ensemble. *Nucleic Acids Research*, 42(22) :e171–e171, 2014.

[186] Jianhua Ruan, Gary D Stormo, and Weixiong Zhang. An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics*, 20(1) :58–66, 2004.

[187] Rajesha Rupaimoole and Frank J Slack. MicroRNA therapeutics : towards a new era for the management of cancer and other diseases. *Nature reviews Drug discovery*, 16(3) :203, 2017.

[188] Jong-hyun Ryu, Sujin Kim, and Hong Wan. Pareto front approximation with adaptive weighted sum method in multiobjective simulation optimization. In *Winter Simulation Conference*, pages 623–633. Winter Simulation Conference, 2009.

[189] Afaf Saaidi, Yann Ponty, and Bruno Sargueil. An integrative approach for predicting the RNA secondary structure for the HIV–1 Gag UTR using probing data. In Hélène Touzet Cédric Lhoussaine, editor, *JOBIM - Journées Ouvertes en Biologie, Informatique et Mathématiques - 2017*, page 102, Lille, France, 2017. URL `https://hal.archives-ouvertes.fr/hal-01534587`.

[190] Yasubumi Sakakibara, Michael Brown, Richard Hughey, I Saira Mian, Kimmen Sjölander, Rebecca C Underwood, and David Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22(23) :5112–5120, 1994.

[191] Raheleh Salari, Rolf Backofen, and S Cenk Sahinalp. Fast prediction of RNA–RNA interaction. *Algorithms for Molecular Biology*, 5(1) :5, 2010.

[192] W Salser. Globin mRNA sequences : analysis of base pairing and evolutionary implications. In *Cold Spring Harbor symposia on quantitative biology*, volume 42, pages 985–1002. Cold Spring Harbor Laboratory Press, 1978.

[193] Abdón Sánchez-Arroyo. Determining the total colouring number is NP–hard. *Discrete Mathematics*, 78(3) :315–319, 1989.

[194] Kengo Sato, Yuki Kato, Michiaki Hamada, Tatsuya Akutsu, and Kiyoshi Asai. IPknot : fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. *Bioinformatics*, 27(13) :i85–i93, 2011.

[195] Cédric Saule and Robert Giegerich. Pareto optimization in algebraic dynamic programming. *Algorithms for Molecular Biology*, 10(1) :22, 2015.

[196] Matthew G Seetin, Wipapat Kladwang, John P Bida, and Rhiju Das. Massively parallel RNA chemical mapping with a reduced bias MAP-seq protocol. In *RNA Folding*, pages 95–117. Springer, 2014.

[197] LS Selvakumar and MS Thakur. Nano RNA aptamer wire for analysis of vitamin B12. *Analytical biochemistry*, 427(2) :151–157, 2012.

[198] Martin J Serra and Douglas H Turner. Predicting thermodynamic properties of RNA. In *Methods in enzymology*, volume 259, pages 242–261. Elsevier, 1995.

[199] Bruce A Shapiro and Kaizhong Zhang. Comparing multiple RNA secondary structures using tree comparisons. *Bioinformatics*, 6(4) :309–318, 1990.

[200] Saad Sheikh, Rolf Backofen, and Yann Ponty. Impact of the energy model on the complexity of RNA folding with pseudoknots. In *Annual Symposium on Combinatorial Pattern Matching*, pages 321–333. Springer, 2012.

[201] Karthik Sindhya, Ankur Sinha, Kalyanmoy Deb, and Kaisa Miettinen. Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In *2009 IEEE Congress on Evolutionary Computation*, pages 2919–2926. IEEE, 2009.

[202] Alok Singh and Ashok Kumar Gupta. A hybrid heuristic for the maximum clique problem. *Journal of Heuristics*, 12(1-2) :5–22, 2006.

[203] Michael F Sloma and David H Mathews. Improving RNA secondary structure prediction with structure mapping data. In *Methods in enzymology*, volume 553, pages 91–114. Elsevier, 2015.

[204] Aleksandar Spasic, Sarah M Assmann, Philip C Bevilacqua, and David H Mathews. Modeling RNA secondary structure folding ensembles using SHAPE mapping data. *Nucleic Acids Research*, 46(1) :314–323, 2017.

[205] Narasimhan Sudarsan, Ming C Hammond, Kirsten F Block, Rüdiger Welz, Jeffrey E Barrick, Adam Roth, and Ronald R Breaker. Tandem riboswitch architectures exhibit complex gene control functions. *Science*, 314(5797) :300–304, 2006.

[206] Yukihiko Sugita, Hideyuki Matsunami, Yoshihiro Kawaoka, Takeshi Noda, and Matthias Wolf. Cryo–EM structure of the Ebola virus nucleoprotein–RNA complex at 3.6 å resolution. *Nature*, 563(7729) :137, 2018.

[207] John Sylva and Alejandro Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158 (1) :46–55, 2004.

[208] John Sylva and Alejandro Crema. A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research*, 180(3) :1011–1027, 2007.

[209] Hakim Tafer and Ivo L Hofacker. RNAplex : a fast tool for RNA–RNA interaction search. *Bioinformatics*, 24(22) :2657–2663, 2008.

[210] Hakim Tafer, Fabian Amman, Florian Eggenhofer, Peter F Stadler, and Ivo L Hofacker. Fast accessibility-based prediction of RNA–RNA interactions. *Bioinformatics*, 27(14) :1934–1940, 2011.

[211] Yin Tang, Emil Bouvier, Chun Kit Kwok, Yiliang Ding, Anton Nekrutenko, Philip C Bevilacqua, and Sarah M Assmann. StructureFold : genome-wide RNA secondary structure mapping and reconstruction in vivo. *Bioinformatics*, 31(16) :2668–2675, 2015.

[212] Michela Taufer, Abel Licon, Roberto Araiza, David Mireles, FHD Van Batenburg, Alexander P Gultyaev, and Ming-Ying Leung. Pseudobase++ : an extension of pseudobase for easy searching, formatting and visualization of pseudoknots. *Nucleic Acids Research*, 37(suppl 1) : D127–D135, 2009.

[213] Ignacio Tinoco, Olke C Uhlenbeck, and Mark D Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293) :362, 1971.

[214] Ignacio Tinoco, Philip N Borer, Barbara Dengler, Mark D Levine, Olke C Uhlenbeck, Donald M Crothers, and Jay Gralla. Improved estimation of secondary structure in ribonucleic acids. *Nature New Biology*, 246(150) :40, 1973.

[215] Brian Tjaden. TargetRNA : a tool for predicting targets of small RNA action in bacteria. *Nucleic Acids Research*, 36(suppl_2) :W109–W113, 2008.

[216] Weitian Tong, Randy Goebel, Tian Liu, and Guohui Lin. Approximating the maximum multiple RNA interaction problem. *Theoretical Computer Science*, 556 :63–70, 2014.

[217] Jung-Fa Tsai, Ming-Hua Lin, and Yi-Chung Hu. Finding multiple solutions to general integer linear programs. *European Journal of Operational Research*, 184(2) :802–809, 2008.

[218] Douglas H Turner and David H Mathews. NNDB : the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Research*, 38 (suppl_1) :D280–D282, 2009.

[219] Douglas H Turner, Naoki Sugimoto, and Susan M Freier. RNA structure prediction. *Annual review of biophysics and biophysical chemistry*, 17(1) :167–192, 1988.

[220] Yasuo Uemura, Aki Hasegawa, Satoshi Kobayashi, and Takashi Yokomori. Tree adjoining grammars for RNA structure prediction. *Theoretical computer science*, 210(2) :277–303, 1999.

[221] Ekunda Lukata Ulungu and Jacques Teghem. The two phases method : An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2) :149–165, 1995.

[222] Jason G Underwood, Andrew V Uzilov, Sol Katzman, Courtney S Onodera, Jacob E Mainzer, David H Mathews, Todd M Lowe, Sofie R Salama, and David Haussler. FragSeq : transcriptome-wide RNA structure probing using high-throughput sequencing. *Nature methods*, 7(12) :995, 2010.

[223] Ilias Vasilopoulos, Varvara G Asouti, Kyriakos C Giannakoglou, and Marcus Meyer. Gradient-based pareto front approximation applied to turbomachinery shape optimization. *Engineering with Computers*, pages 1–11, 2019.

[224] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416, 2007.

[225] Amy E Walter, Douglas H Turner, James Kim, Matthew H Lyttle, Peter Müller, David H Mathews, and Michael Zuker. Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proceedings of the National Academy of Sciences*, 91(20) :9218–9222, 1994.

[226] Yue Wan, Kun Qu, Zhengqing Ouyang, and Howard Y Chang. Genome-wide mapping of RNA structure using nuclease digestion and high-throughput sequencing. *Nature protocols*, 8(5) : 849, 2013.

[227] Yue Wan, Kun Qu, Qiangfeng Cliff Zhang, Ryan A Flynn, Ohad Manor, Zhengqing Ouyang, Jiajing Zhang, Robert C Spitale, Michael P Snyder, Eran Segal, et al. Landscape and variation of RNA secondary structure across the human transcriptome. *Nature*, 505(7485) :706, 2014.

[228] Linyu Wang, Yuanning Liu, Xiaodan Zhong, Haiming Liu, Chao Lu, Cong Li, and Hao Zhang. DMfold : A novel method to predict RNA secondary structure with pseudoknots based on deep learning and improved base pair maximization principle. *Frontiers in genetics*, 10 :143, 2019.

[229] Jeffrey S Warren and Illya V Hicks. Combinatorial branch-and-bound for the maximum weight independent set problem. *Relatório técnico, Texas A&M University, Citeseer*, 2006.

[230] Stefan Washietl, Ivo L Hofacker, Peter F Stadler, and Manolis Kellis. RNA folding with soft constraints : reconciliation of probing data and thermodynamic secondary structure prediction. *Nucleic Acids Research*, 40(10) :4261–4272, 2012.

[231] Michael S Waterman. Secondary structure of single-stranded nucleic acids. *Adv. math. suppl. studies*, 1 :167–212, 1978.

[232] Joseph M Watts, Kristen K Dang, Robert J Gorelick, Christopher W Leonard, Julian W Bess Jr, Ronald Swanstrom, Christina L Burch, and Kevin M Weeks. Architecture and secondary structure of an entire HIV-1 RNA genome. *Nature*, 460(7256) :711, 2009.

[233] Kevin A Wilkinson, Robert J Gorelick, Suzy M Vasa, Nicolas Guex, Alan Rein, David H Mathews, Morgan C Giddings, and Kevin M Weeks. High-throughput SHAPE analysis reveals structures in HIV-1 genomic RNA strongly conserved across distinct biological states. *PLoS biology*, 6 (4) :e96, 2008.

[234] Cindy L Will and Reinhard Lührmann. Spliceosome structure and function. *Cold Spring Harbor perspectives in biology*, 3(7) :a003707, 2011.

[235] H Paul Williams. *Model building in mathematical programming*. Wiley, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, 1999.

[236] KELLY P Williams and DAVID P Bartel. Phylogenetic analysis of tmRNA secondary structure. *Rna*, 2(12) :1306–1310, 1996.

[237] Arthur L Williams Jr and Ignacio Tinoco Jr. A dynamic programming algorithm for finding alternative RNA secondary structure. *Nucleic Acids Research*, 14(1) :299–315, 1986.

[238] John L Woolford and Susan J Baserga. Ribosome biogenesis in the yeast saccharomyces cerevisiae. *Genetics*, 195(3) :643–681, 2013.

[239] Qinghua Wu, Jin-Kao Hao, and Fred Glover. Multi-neighborhood tabu search for the maximum weight clique problem. *Annals of Operations Research*, 196(1) :611–634, 2012.

[240] Yang Wu, Binbin Shi, Xinqiang Ding, Tong Liu, Xihao Hu, Kevin Y Yip, Zheng Rong Yang, David H Mathews, and Zhi John Lu. Improved prediction of RNA secondary structure by integrating the free energy model with restraints derived from experimental probing data. *Nucleic Acids Research*, 43(15) :7247–7259, 2015.

[241] Yang Wu, Rihao Qu, Yiming Huang, Binbin Shi, Mengrong Liu, Yang Li, and Zhi John Lu. RNAex : an RNA secondary structure prediction server enhanced by high-throughput structure-probing data. *Nucleic Acids Research*, 44(W1) :W294–W301, 2016.

[242] Stefan Wuchty, Walter Fontana, Ivo L Hofacker, and Peter Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers : Original Research on Biomolecules*, 49(2) :145–165, 1999.

[243] Tianbing Xia, John SantaLucia Jr, Mark E Burkard, Ryszard Kierzek, Susan J Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson–Crick base pairs. *Biochemistry*, 37(42) :14719–14735, 1998.

[244] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2) :165–193, 2015.

[245] Xiaojun Xu and Shi-Jie Chen. VfoldCPX server : Predicting RNA–RNA complex structure and stability. *PloS One*, 11(9) :e0163454, 2016.

[246] Kenji Yamamoto and Hiroshi Yoshikura. An improved algorithm for the prediction of optimum and suboptimum folding structures of long single-stranded RNA. *Bioinformatics*, 3(1) : 31–35, 1987.

[247] Huanwang Yang, Fabrice Jossinet, Neocles Leontis, Li Chen, John Westbrook, Helen Berman, and Eric Westhof. Tools for the automatic identification and classification of RNA base pairs. *Nucleic Acids Research*, 31(13) :3450–3460, 2003.

[248] Joseph N Zadeh, Conrad D Steenberg, Justin S Bois, Brian R Wolfe, Marshall B Pierce, Asif R Khan, Robert M Dirks, and Niles A Pierce. NUPACK : analysis and design of nucleic acid systems. *Journal of computational chemistry*, 32(1) :170–173, 2011.

[249] Kourosh Zarringhalam, Michelle M Meyer, Ivan Dotu, Jeffrey H Chuang, and Peter Clote. Integrating chemical footprinting data into RNA secondary structure prediction. *PLoS One*, 7 (10) :e45160, 2012.

[250] Feng Zhang, Sébastien Lemieux, Xiling Wu, Daniel St Arnaud, Cynthia T McMurray, François Major, and Dwight Anderson. Function of hexameric RNA in packaging of bacteriophage φ29 DNA in vitro. *Molecular cell*, 2(1) :141–147, 1998.

[251] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6) :1245–1262, 1989.

[252] Weihua Zhang and Marc Reimann. A simple augmented ϵ-constraint method for multi-objective mathematical integer programming problems. *European Journal of Operational Research*, 234(1) :15–24, 2014.

[253] Xiaofeng Zhang, Chuangye Yan, Jing Hang, Lorenzo I Finci, Jianlin Lei, and Yigong Shi. An atomic structure of the human spliceosome. *Cell*, 169(5) :918–929, 2017.

[254] Yi Zhao, Hui Li, Shuangsang Fang, Yue Kang, Wei Wu, Yajing Hao, Ziyang Li, Dechao Bu, Ninghui Sun, Michael Q Zhang, et al. NONCODE 2016 : an informative and valuable data source of long non-coding RNAs. *Nucleic Acids Research*, 44(D1) :D203–D208, 2015.

[255] Michael Zuker. Computer prediction of RNA structure. In *Methods in enzymology*, volume 180, pages 262–288. Elsevier, 1989.

[256] Michael Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244(4900) : 48–52, 1989.

[257] Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13) :3406–3415, 2003.

[258] Michael Zuker and David Sankoff. RNA secondary structures and their prediction. *Bulletin of mathematical biology*, 46(4) :591–621, 1984.

[259] Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1) :133–148, 1981.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Prédiction de structures secondaires d'ARN et de complexes d'ARN avec pseudonoeuds - Approches basées sur la programmation mathématique multi-objectif

Mots clés : Bio-informatique des ARN / Prédiction de structures secondaires / Pseudonœud / Complexe d'ARN / Optimisation multi-objectif / Programmation linéaire

Résumé : Dans cette thèse, nous proposons de nouveaux algorithmes et outils pour la prédiction de structures secondaires d'ARN et de complexes d'ARN, incluant des motifs particuliers, difficiles à prédire, comme les pseudonœuds. La prédiction de structures d'ARN reste une tâche difficile, et les outils existants, pourtant nombreux, ne donnent pas toujours de bonnes prédictions.
Afin de prédire plus précisément ces structures, nous proposons ici des algorithmes qui : i) prédisent les k-meilleures structures; ii) combinent plusieurs modèles de prédiction, afin de bénéficier des avantages de chacun; iii) sont capables de prendre en compte des contraintes utilisateurs et des données biologiques structurales telles que le SHAPE.
Nous avons développé trois outils: BiokoP pour la prédiction de structures secondaires d'un ARN, et RCPred et C-RCPred pour la prédiction de structures secondaires de complexes d'ARN.
L'outil BiokoP propose plusieurs structures optimales et sous-optimales grâce à la combinaison de deux modèles de prédiction, le modèle énergétique MFE et le modèle probabiliste MEA. Cette combinaison est réalisée grâce à la programmation mathématique multi-objectif, où chaque modèle est assimilé à une fonction objectif. À cet effet, nous avons développé un algorithme générique retournant les k-meilleures courbes de Pareto d'un programme linéaire en nombres entiers bi-objectif.
L'outil RCPred, basé sur le modèle MFE, propose plusieurs structures sous-optimales. Il tire parti des nombreux outils existants pour la prédiction de structures secondaires d'ARN seuls et d'interactions ARN-ARN, en prenant en compte des structures secondaires et interactions déjà prédites en entrée. L'objectif de RCPred est de trouver les meilleures combinaisons possibles parmi ces entrées.
L'outil C-RCPred est une nouvelle version de RCPred, prenant en compte des contraintes utilisateurs et des données biologiques structurales (SHAPE, PARS et DMS). C-RCPred est basé sur un algorithme multi-objectif, où les différents objectifs correspondent au modèle MFE, au respect des contraintes utilisateurs et à l'accord avec les données biologiques structurales.

Title : RNA and RNA complex secondary structure prediction with pseudoknots - Multi-objective mathematical programming based approaches

Keywords : RNA bioinformatics / Secondary structure prediction / Pseudoknot / RNA complex / Multi-objective optimization / Linear programming

Abstract : In this thesis, we propose new algorithms and tools to predict RNA and RNA complex secondary structures, including particular RNA motifs, difficult to predict, like pseudoknots. RNA structure prediction stays a difficult task, and the numerous existing tools don't always give good predictions.
In order to predict structures that are as close as possible to the real ones, we propose to develop algorithms that: i) predict the k-best structures; ii) combine several models of prediction to take advantage of each; iii) are able to take into account user constraints and structural data like SHAPE.
We developed three tools: BiokoP for predicting RNA secondary structures and RCPred and C-RCPred for predicting RNA complex secondary structures.
The tool BiokoP proposes several optimal and sub-optimal structures thanks to the combination of two prediction models, the energy model MFE and the probabilistic model MEA. This combination is done with multi-objective mathematical programming, where each model is associated to an objective function. To this end, we developed a generic algorithm returning the k-best Pareto curves of a bi-objective integer linear program.
The tool RCPred, based on the MFE model, proposes several sub-optimal structures. It takes advantage of the numerous existing tools for RNA secondary structure prediction and for RNA-RNA interaction prediction, by taking as input predicted secondary structures and RNA-RNA interactions. The goal of RCPred is to find the best combination among these inputs.
The tool C-RCPred is a new version of RCPred, taking into account user constraints and structural data (SHAPE, PARS, DMS). C-RCPred is based on a multi-objective algorithm, where the different objectives are the MFE model, the fulfillment of the user constraints and the concordance with the structural data.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France