Modèle de scaffolding

Étude bibliographique

Master *Sciences et Technologies*,

Mention *Informatique*,

Parcours Master Informatique Théorique

Auteur
Clément Dallard

Superviseurs
Annie Chateau
Rodolphe Giroudeau

Lieu de stage
LIRMM UM5506 - CNRS, Université de Montpellier

version 1 22 mars 2016

Résumé

Le séquençage de données génomiques nécessite plusieurs traitements informatiques qui doivent manipuler de grandes quantités de données en un temps raisonnable. Parmi ces étapes, le processus d'*échafaudage de génome* (ou *scaffolding*) cherche à trouver une orientation et un ordre sur des fragments d'`ADN` pour représenter au mieux les génomes dont ils sont issus. Ce problème est NP-complet, et il est donc nécessaire de concevoir des heuristiques efficaces en terme de temps de calcul pour le résoudre, et si possible en ayant une garantie de performance sur le résultat obtenu. Ce rapport présentera le problème de *scaffolding* en introduisant son contexte biologique et dressera un état de l'art recouvrant diverses modélisations informatiques. La problématique du stage y sera définie et les objectifs déterminés.

Table des matières

Chapitre 1

Introduction

L'*échafaudage de génome*, ou *scaffolding*, est une des étapes du processus de production de séquences génomiques. Après avoir *séquencé* et *assemblé* les molécules d'ADN, on obtient des fragments d'ADN qu'il nous faut traiter : les *contigs*. L'étape de *scaffolding* essaie de les ordonner et de les orienter de façon à reconstituer l'entièreté du génome[1] . Les *contigs* ainsi orientés et ordonnés entre eux sont appelés *scaffolds*.

Le génome est l'ensemble du patrimoine génétique d'un individu, ou d'une espèce, codé dans son ADN. Il contient l'ensemble de l'information héréditaire présente dans chaque cellule de chaque organisme vivant. La parabole la plus utilisée est de comparer le génome à une encyclopédie dont les chromosomes seraient les volumes, les gènes les phrases, et le tout serait écrit sur un alphabet de quatre lettres (ou bases) : {A,T,G,C}[2].

Le premier intérêt du séquençage du génome est médical. Par exemple, mieux comprendre le fonctionnement d'un organisme pathogène pour mieux le combattre, ou repérer si un individu est porteur d'une maladie héréditaire... On peut aussi penser aux avancées possibles en biotechnologie, en agronomie ou dans l'industrie, et bien entendu son utilité pour la police. Mais, l'intérêt est aussi scientifique. Découvrir, comprendre, comparer et explorer le vivant comme jamais il n'a été possible auparavant. Avec le séquençage, la porte de la *génomique*[3] s'est ouverte et l'histoire du vivant, par la *phylogénie*, s'écrit devant nous.

Les dernières avancées technologiques ont permis un séquençage nettement moins coûteux que les techniques précédemment utilisées. La technologie HTS (High-Throughput Sequencing : [24]) rend le séquençage beaucoup plus accessible en terme de coût et permet d'obtenir rapidement des résultats. Durant la dernière décennie, d'importantes quantités de génomes ont été séquencées grâce à cette technologie, et une abondance de séquences d'ADN est disponible dans les bases de données[4]. Pour autant, la plupart des récents projets de séquençage de génomes ne sont pas terminés, car le goulot d'étranglement de ce processus est informatique : le traitement de ces données prend du temps... La plupart des techniques de séquençage produisent des paires de *reads*. Un

[1] illustration du processus de séquençage en Annexe A figure A.1

[2] adénine, thymine, guanine, et cytosine

[3] science qui étudie le génome

[4] https ://www.ncbi.nlm.nih.gov : plus de 80000 génomes d'organismes et 5×10^{15} nucléotides

read est l'extrémité d'un fragment d'ADN dont la séquence est connue, et une paire de *reads* délimite le début et la fin d'un même fragment. Il faut *assembler* ces *reads* pour obtenir des *contigs*, c'est à dire des séquences du génome. Les séquenceurs[5] qui utilisent la technologie HTS fragmentent beaucoup ces données, en ne séquençant que de courts *reads*, autrement dit la quantité de *reads* obtenue sera élevée par rapport à celle recueillie via d'autres technologies, rendant d'autant plus laborieuses les tâches d'*assemblage* et de *scaffolding*. La motivation à accélérer le processus de reconstitution du génome anime la communauté scientifique, et nous y participons en proposant des solutions au problème de *scaffolding*.

Le problème de *scaffolding* a été formalisé et classé comme NP-complet par *Huson et al.* dans [14]. Une solution optimale est censée représenter au mieux la façon dont le génome est réellement agencé. Aujourd'hui, plusieurs modélisations et méthodes existent pour le résoudre, et nous les présenterons dans les deux premiers chapitres de ce rapport.

De par la complexité du problème, la majorité des outils le résolvent via une approche heuristique. Ces approches, bien que très répandues, ne permettent pas d'évaluer l'optimalité de la solution renvoyée qui peut-être arbitrairement éloignée de la solution optimale[6]. D'autres outils adoptent une résolution exacte, mais aucun passage à l'échelle sur des génomes existants n'est alors envisageable. Une autre approche intéressante est de chercher des algorithmes d'approximation polynomiaux de façon à résoudre le problème en un temps acceptable tout en ayant une garantie sur l'optimalité de la solution retournée. Comme démarche parallèle, étudier de façon théorique la complexité du problème sur des classes de graphes particulières permet la conception d'algorithmes exacts efficaces sur certains types de données biologiques.

Dans cette étude bibliographique, nous présenterons en détails le problème de *scaffolding* d'un point de vue informatique en énumérant les résultats théoriques associés. Différentes modélisations seront proposées et comparées, ce qui nous permettra d'exposer plusieurs techniques de résolution au travers l'étude des principaux outils disponibles. Nous terminerons par la présentation de notre propre outil, encore en développement, et des ambitions quant à sa réalisation.

[5] appareils qui automatisent l'opération de séquençage de l'ADN, en déterminant l'ordre des bases

[6] nous verrons néanmoins que les solutions sont généralement très bonnes

Chapitre 2

Modélisations du problème

2.1 Préambule

Tout problème réel résolu par ordinateur est intrinsèquement lié à sa modélisation en machine. Le problème de *scaffolding* n'échappe pas à cette règle, et nous présenterons différentes modélisations que nous retrouverons lors de la présentation des outils existants dans le prochain chapitre. Il existe des outils qui se basent sur un génome de référence pour résoudre le problème de *scaffolding*, mais on considère ici des méthodes qui n'en disposent pas. On parle dans ce cas d'approche *de novo*.

Définition 1. *Les contigs sont les fragments de génome qu'il faut orienter et ordonner de façon à reconstituer le génome entier. Ces fragments sont des séquences d'*`ADN` *dont on connaît la longueur et le code génétique les constituant. Plus formellement, ce sont des mots de longueur variable sur l'alphabet* $\mathcal{A}^*$ *avec* $\mathcal{A} = \{A, C, G, T\}$.

Définition 2. *Un scaffold est un ensemble de* contigs *orientés et consécutifs formant une séquence génomique linéaire ou cyclique.*

Définition 3. *Un support entre deux contigs est un paire de* reads *qui possède une extrémité chevauchant un* contig *et l'autre extrémité chevauchant un deuxième* contig.

La notion de chevauchement est à appréhender au sens large, et un *read* chevauchant un *contig* peut correspondre à une partie interne du *contig*. Dans la suite du document, nous appellerons *graphe de scaffold* le graphe qui représente la collection de *contigs* à orienter et ordonner, reliés par des supports. Dans un effort de lisibilité nous avons regroupé, quand cela était possible, les différentes modélisations que proposent les outils en une modélisation équivalente.

2.2 Modélisation classique

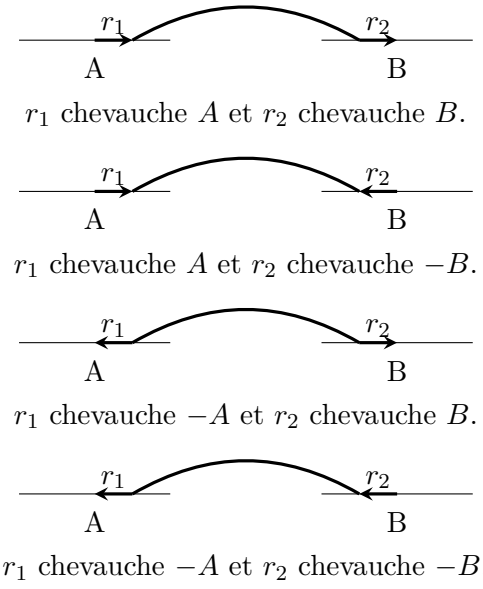
Une modélisation répandue est de considérer un *bidirected* graphe dont les sommets sont des *contigs* et les arêtes les reliant sont pondérées en fonction du nombre de supports que partagent les deux *contigs* en extrémité.

Définition 4. *Un bidirected graphe est un graphe dont les arêtes sont orientées indépendamment en chacune de leurs extrémités. Une extrémité d'arête peut être orientée vers l'intérieur ou l'extérieur, ce qui peut définir trois types d'arêtes différentes.*

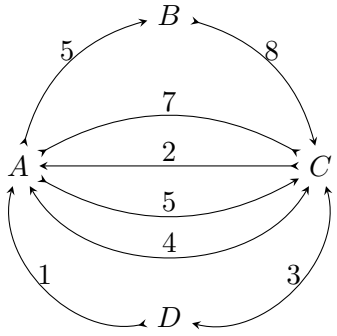
Les pondérations des arêtes sont calculées avant l'étape du *scaffolding*, et après l'étape d'assemblage[1]. Cette étape intermédiaire qui pondère les arêtes nécessite un *mapping*.

Définition 5. *Le mapping est l'opération d'alignement des* reads *sur les* contigs. *De cette opération, on déduit les arêtes* inter-contigs *(ou arcs suivant le type de graphe) et leurs poids en fonction du nombre de supports que partagent les* contigs *en extrémité.*

On considère qu'une orientation d'un *contig* définit son sens de lecture. On remarque que deux *contigs* c_1 et c_2 peuvent être reliés ensemble de quatre manières différentes selon le sens de lecture des *reads* qui forment un support les connectant. Soit (r_1, r_2) une paire de *reads* qui est un support entre deux contigs c_1 et c_2 telle que r_1 chevauche c_1 et r_2 chevauche c_2. En fonction du sens de lecture de r_1 par rapport à l'orientation de c_1, et de celui de r_2 par rapport à l'orientation de c_2, on peut obtenir jusqu'à quatre configurations différentes, puisqu'il existe deux sens possibles pour r_1 et de même pour r_2 (voir figure 2.1a où les contigs A et C partagent tous les types d'arêtes possibles). Ainsi, pour traduire l'information sur l'orientation des *contigs* l'un par rapport à l'autre, on utilise un *bidirected* graphe (voir figure 2.1b).



(a) Chevauchements possibles d'un support (r_1, r_2) entre deux contigs A et B



(b) Exemple de *bidirected* graphe pondéré

FIGURE 2.1 – Arêtes et chevauchements *inter-contigs*.

[1]étape qui consiste à trouver la meilleure façon d'assembler les fragments d'`ADN` en de plus longues séquences en optimisant leur chevauchement (généralement modélisée par le problème de la *plus courte super-chaîne* [27])

Alors, le choix d'une arête pour relier deux sommets fixe à la fois l'ordre des *contigs* et leurs orientations (suivant le type d'arête). Toute orientation et tout ordre sur les *contigs* peuvent être représentés par une collection de chemins sommets disjoints sur le *graphe de scaffold* ainsi modélisé. On peut aussi autoriser les cycles sommets disjoints, ce qui se traduit biologiquement par des chromosomes cycliques. Le problème de *scaffolding* peut alors se traduire comme un problème d'optimisation où l'on cherche une collection de chemins et cycles sommets disjoints qui maximise la somme des poids des arêtes empruntés. Cette modélisation peut paraître lourde mais n'est généralement pas directement utilisée mais d'abord simplifiée heuristiquement en un graphe orienté $G = (V, A)$ avant la résolution, comme nous le verrons dans le chapitre suivant.

2.3 Modélisation avec couplage parfait

Une deuxième modélisation, plus récente, proposée dans [3][2], représente le *graphe de scaffold* comme un graphe non-orienté, sans boucle, où l'on connaît un couplage parfait.

Définition 6. *Un couplage parfait est un ensemble d'arête d'un graphe formant un couplage tel que tout sommet du graphe est incident à exactement une arête du couplage.*

Les arêtes du couplage représentent les *contigs*. Ici aussi le *scaffolding* se traduit comme un problème d'optimisation sur le *graphe de scaffold* qui cherche à maximiser la somme des poids des arêtes empruntées par un ensemble de cycles et chemins sommets disjoints, dont les arêtes sont alternativement dans le couplage et en dehors. Cette représentation évite notamment l'utilisation d'un *bidirected* graphe. Un raffinement supplémentaire a été proposé avec cette modélisation par l'ajout de contraintes sur le nombres de chemins et cycles que peut contenir une solution.

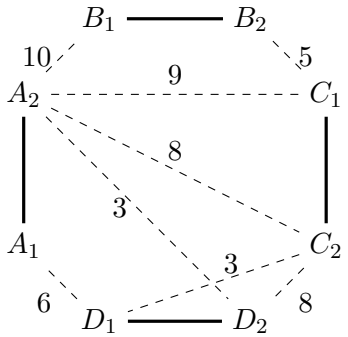


FIGURE 2.2 – Représentation des *contigs* comme arêtes du couplage, et arêtes *inter-contigs* pondérées.

[2]cette modélisation a déjà été utilisée dans des outils comme *BESST* ou *SCARPA* sans jamais être comparée à un graphe dont on connaît un couplage parfait

2.4 Modélisations alternatives

Consecutive-Ones Property

Une autre modélisation proposée dans [5] considère le problème de reconnaissance de *Consecutive-Ones Property (C1P)*[21] dans des matrices codant des hypergraphes. Une matrice codant un hypergraphe $H = (V, E)$ à $|V| = n$ sommets et $|E| = m$ arêtes est une matrices binaire $\mathcal{A}_{n,m}$ telle que $a_{i,j} \in \mathcal{A} = 1$ si et seulement si $v_i \in E_j$, avec E_j un sous-ensemble de sommets. Une matrice binaire possède une *C1P* quand il existe une permutation de ses colonnes (respectivement lignes) qui rend les *1* consécutifs dans toutes ses lignes (respectivement colonnes). Les auteurs ont de plus pris en compte la possible répétition de même *contigs* (on parle de multiplicité de *contigs*) dans la solution et présentent d'intéressants résultats d'algorithmes d'approximation du problème de reconnaissance de *scaffolding* dans le cas où il n'y a pas de contraintes sur le nombre de chemins et cycles. Cette modélisation néanmoins n'a jusque là pas donné lieu à un outil spécifique.

Mixed Integer Programming

En plus de ces modélisations qui traduisent les données sur les *contigs* par la création d'un graphe pondéré, une approche par programmation mathématique à variables mixte[3] est aussi utilisée.

Définition 7. *Un programme mathématique à variables mixtes (Mixed Integer Program) est un problème sous forme d'équations et d'inéquations sur des variables entières ou réelles (donc mixtes) où l'on cherche à minimiser, ou maximiser, une fonction (appelée fonction objectif) tout en satisfaisant les contraintes imposées par les inéquations.*
La programmation mathématique à variables mixtes (Mixed Integer Programming ou MIP) permet de résoudre ce genre de problème.

Ces approches offrent généralement de bons résultats sur de petits génomes, mais de part le fait qu'ils ne peuvent résoudre le problème que de manière exacte, leur complexité empêche tout passage à l'échelle.

Chaque modélisation propose ses avantages, et nous verrons que les outils spécialisés dans la résolution du *scaffolding* les utilisent de manières successives, ou combinées. Il faut savoir passer d'un paradigme à l'autre pour espérer obtenir des résultats satisfaisants, tant sur le plan de l'optimalité que celui du temps de calcul. Le prochain chapitre entend faire un état de l'art des différentes méthodes utilisées par les principaux outils de *scaffolding*.

[3] pour plus de détails sur les techniques de résolution de problèmes à variables mixtes, se référer à [2]

Chapitre 3

Méthodes et techniques de résolution

Nous présenterons dans ce chapitre les outils les plus connus pour le *scaffolding*, le but étant de comparer les approches et méthodes de résolution de ces outils d'un point de vue théorique.

Sauf si précisé, nous considérerons dans la suite de ce rapport qu'une arête entre deux *contigs* est pondérée en fonction du nombre de supports qui les lient. L'étape de *mapping*, préliminaire à celle de *scaffolding*, est essentielle à l'obtention de résultats proches de l'optimal. En effet, on observe que selon la technique (ou l'outil) de *mapping* utilisée les résultats sont sensiblement différents, comme on peut le constater dans [13] qui propose une évaluation de plusieurs outils de *scaffolding* sur divers jeux de données.

3.1 Approche gloutonne

Définition 8. *Un algorithme glouton (greedy en anglais) est un algorithme qui n'envisage qu'un seul choix améliorant sa solution actuelle à chaque étape.*

Pour la majorité des problèmes, un algorithme glouton est une heuristique et donc ne permet pas d'atteindre un optimum global dans tous les cas, mais au contraire peut renvoyer un optimal local. Les avantages de ces algorithmes sont évidemment leur faible complexité en temps, ce qui leur permet d'être utilisables sur de grandes instances, et leur facilité d'implémentation. Dans le cas du *scaffolding*, ces algorithmes permettent d'obtenir rapidement une solution vis à vis d'autres techniques, mais celle-ci peut être très éloignée de la solution optimale.

Des outils comme *Bambus*[1] ([23]) emploient ce paradigme glouton pour résoudre le *scaffolding*. Après la première étape de *mapping*, *Bambus* utilise un algorithme glouton qui connecte ensemble les *contigs* partageant le plus de liens, et ignore par la suite toute arête ne respectant pas l'ordre ou l'orientation des *scaffolds* déjà créés.

[1] *Bambus* a été le premier outil autonome développé pour le *scaffolding*

SSPACE ([1]) aussi emploie ce paradigme glouton mais l'étape de *mapping* est gérée par un programme déjà existant, *Bowtie* ([19]). Plusieurs bibliothèques de *reads* peuvent être utilisées pour améliorer et compléter les données sur les *contigs*. Une simplification, héritée de l'outil *SSAKE* ([28]), ne conservera que les liens qui possèdent un poids supérieur à un seuil fixé par l'utilisateur. Remarquons que *SSPACE* et *SSAKE* utilisent une structure de table de hachage pour résoudre le *scaffolding*. Les *contigs* ainsi pairés sont liés entre eux (pour former les *scaffolds*) de manière itérative, en commençant par le *contig* de plus grande taille, dès lors qu'un poids suffisant sur le lien conforte l'idée que les deux *contigs* sont successifs dans la solution optimale. Si plusieurs choix sont possibles quant à l'orientation d'un *contig*, une heuristique vise à choisir la plus plausible localement en calculant un score. L'extension d'un *scaffold* s'arrête dès qu'on ne peut plus lier de *contig* ou que le score n'est pas suffisant.

SSAKE utilise lui aussi une méthode gloutonne, mais les *contigs* sont reliés selon un critère basé sur la notion de *k-mer*.

Définition 9. *Un k-mer est une sous-chaîne de taille k dans une chaîne.*

Par exemple, l'ensemble des *4-mers* de la chaîne $m = ACCTGAT$ est $\{ACCT, CCTG, CTGA, TGAT\}$.

Ces trois outils sont à notre connaissance les seuls à utiliser une approche strictement gloutonne pour la résolution du problème *scaffolding*. Leur intérêt est leur faible complexité, ce qui les rends très performants en temps. Pour autant, il est évident que la qualité de la solution peut-être déplorable sur certaines données[2], d'abord par les simplifications effectuées qui ne garantissent pas que l'instance réduite est équivalente à l'instance initiale, ensuite par l'aspect local de la recherche de maximum.

3.2 Utilisation d'un graphe

Nous avons présenté dans le chapitre 2 des modélisation qui s'appuient sur un graphe. Ces modélisations sont les plus répandues parmi les outils de *scaffolding* car elles permettent d'appliquer beaucoup de méthodes, algorithmes et heuristiques hérités de la théorie des graphes ou de la recherche opérationnelle. De plus, la représentation sous forme de graphe permet de faire ressortir d'éventuelles structures dans les données qui peuvent être exploitées pendant la résolution pour améliorer la solution ou réduire les complexités des algorithmes employés. Nous citerons dans cette partie les outils qui utilisent principalement cette modélisation dans la résolution.

[2] remarquons tout de même que selon [13], *SSPACE* est un des outils les plus performants en pratique

Par reconnaissance de structures

Bambus 2([18]), à ne pas confondre avec *Bambus* qui n'est pas simplement une première version, utilise une représentation sous forme de graphe et profite de sa structure pour repérer les multiplicités de *contigs* (*contigs* qui apparaissent de manière répétée dans les données). Le graphe est alors un *bidirected* graphe comme présenté dans le chapitre précédent. Les multiplicités de *contigs* apparaissent très fréquemment en *métagénomique*[3]. Ces multiplicités peuvent exprimer deux informations sensiblement différentes : soit les *contigs* sont effectivement les mêmes et on peut les contracter en un seul, soit ils appartiennent à deux parties distinctes d'un ou plusieurs génomes et ils doivent être considérés séparément. *Bambus 2* utilise le graphe pour repérer des structures qui représentent ces multiplicités et permet de décider comment traiter les *contigs* les constituant dans le *graphe de scaffold*. Les structures recherchées sont des ensembles de chemins sommets disjoints qui partagent une source et un puits communs. En cherchant de tels ensembles, et en bornant la longueur des chemins en terme de longueur de séquence (longueur de la chaîne de caractères que représente le chemin), on peut comparer ces chemins et décider s'il y a multiplicité ou s'ils sont trop éloignés pour être agrégés. Une orientation est déduite par une heuristique ([17]) en temps $O(|V| + |E|)$, avec V l'ensemble des sommets E l'ensemble des arêtes du graphe, qui cherche à minimiser le nombre d'arêtes à enlever dans le *bidirected* graphe pour obtenir un graphe orienté[4] ; ce problème est équivalent au *problème de sous-graphe biparti maximal* qui est NP-difficile ([11]). Le placement des *contigs*, c'est à dire l'ordre dans lesquels ils seront agencés pour former des *scaffolds*, est lui aussi déduit en temps $O(|V| + |E|)$ en deux étapes (la première calcule un ordre partiel sur les contigs, la deuxième simplifie le graphe en supprimant les arêtes transitives). Une dernière étape permet d'identifier les variations [5] dans le graphe orienté acyclique ainsi obtenu en recherchant des chemins orientés de même longueurs qui partagent la même source et le même puits. *Bambus 2* utilise donc intensément les propriétés et résultats de cette modélisation sous forme de graphe.

Par simplification

SCARPA ([8]) utilise la théorie des graphes pour répondre au problème de *scaffolding* en proposant un algorithme FPT pour trouver une orientation qui maximise le nombre d'arêtes conservées.

Définition 10. *Un algorithme de complexité FPT (Fixed-Parameter Tractable, ou résoluble à paramètre fixé) est un algorithme qui permet de résoudre un problème paramétré par k en un temps $f(k) \cdot poly(n)$ où f est une fonction quelconque et poly une fonction polynomiale.*

[3] procédé qui étudie le contenu génétique d'un échantillon trouvé dans la nature (océan, intestin, air, sols) renfermant plusieurs organismes dont les génomes seront séquencés ensemble

[4] l'heuristique est une *2-approximation* du nombre minimal d'arcs à enlever

[5] en métagénomique, cela correspondrait à des organismes proches génétiquement, mais dont on peut remarquer des différences sur une partie de leurs génomes respectifs

Le *graphe de scaffold* correspond ici à la modélisation des *contigs* par un couplage parfait. En suivant le même principe que celui présenté dans le chapitre 2, un graphe non-orienté est construit et une détection des cycles impairs est réalisée. Un cycle impair se traduit par une non-alternance entre les arêtes du couplage et les arêtes hors couplage dans le cycle, et comme deux arêtes du couplage ne peuvent pas être consécutives alors il existe forcément deux arêtes hors du couplage consécutives, ce qui ne peut représenter un *scaffold*. Il faut donc retirer du graphe un ensemble d'arêtes pour *casser* les cycles impairs (rendre le graphe biparti) et obtenir un graphe qui correspond à une orientation des *contigs*. En imposant l'orientation d'un *contig* on impose l'orientation de tous les *contigs* qui appartiennent à la même composante connexe, ce qui n'est pas le cas dans un graphe où il existe des cycles impairs. Alors que *Bambus* proposait un algorithme *2-approché* pour trouver le nombre minimal d'arcs à enlever, *SCARPA* utilise un algorithme exact ([20]) en $O(3^k kmn)$ avec n le nombre de sommets[6], m le nombre d'arêtes et k le nombre maximum d'arêtes à retirer. Nous précisons que bien que l'algorithme soit exact, son utilisation reste une heuristique dans le sens où l'algorithme peut retirer un sous-ensemble d'arêtes nécessaire à la réalisation d'un *scaffolding* maximum. De plus, l'algorithme retire éventuellement des *contigs* du graphe (seulement s'il est strictement plus rentable d'enlever un *contig* que de retirer une arête hors du couplage), ce qui permet de retirer du graphe des *contigs* mal assemblés et qui ont peu de support. Le graphe sans cycle impair est alors transformé en un graphe orienté par contraction des arêtes du couplage (voir figure 3.1).

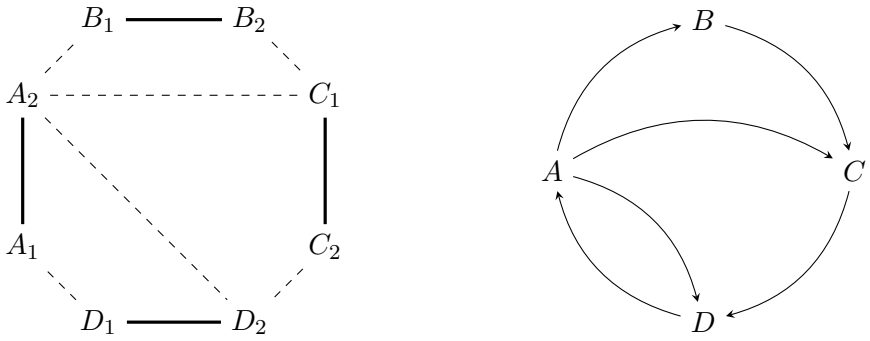


FIGURE 3.1 – Dans un graphe sans cycle de longueur impaire, contraction des arêtes du couplage, représentées en gras, pour obtenir un graphe orienté.

SCARPA, ne prenant pas en compte les *scaffolds* circulaires (des cycles dont les sommets sont des *contigs*), utilise une heuristique ([9]) en temps $O(m)$ pour résoudre le problème NP-difficle de *Feedback arc set* ([16]) qui cherche à enlever un nombre minimum d'arcs à un graphe orienté pour obtenir un graphe sans cycles (orientés). Le graphe orienté acyclique peut alors être considéré comme un ordre (topologique) sur les sommets du graphe, donc sur les *contigs*. Cet ordre peut ne pas être unique et une dernière étape réalisée par programmation en nombres entiers fabrique les *scaffolds* en reliant les *contigs* ensemble.

[6]la transformation qui permet d'utiliser l'algorithme peut, dans le pire des cas, augmenter de $2 \times m$ le nombre de sommets de départ

Par contraction

Un troisième outil, *OPERA* ([10]), utilise une méthode par contraction du *graphe de scaffold*. Le graphe est simplifié pour faire émerger les propriétés nécessaires à la contraction du graphe qu'un algorithme de programmation dynamique[7] utilisera pour résoudre l'instance. Ce graphe simplifié est alors résolu de manière exacte. *OPERA* ne permet pas de prendre en compte les multiplicités de *contigs*, mais elles peuvent néanmoins être repérées par un outils tiers avant l'appel d'*OPERA* sans que cela n'empêche la résolution par *OPERA*.

Par statistiques

SOPRA ([6]) quant à lui aborde le problème par optimisation statistique. Après l'étape de *mapping*, *SOPRA* retire du graphe les *contigs* considérés comme inconsistants (ou chimériques) et qui seraient dus à des erreurs de séquençage ou de *mapping*. Ensuite, le programme utilise un algorithme pour trouver une orientation des *contigs* qui tente de maximiser le poids des arcs restants, et la position des *contigs* (leur ordre) est définie. L'orientation et l'ordre sont déterminés par des heuristiques qui utilisent des données statistiques sur la répartition des supports entre les *contigs*.

BESTT ([25]) aussi utilise une approche statistique, bien que différente de celle de *SOPRA*, en commençant par relier les *contigs* les plus longs en premier. Une détection des liens et *contigs* inconsistants est aussi opérée. Il peut aussi utiliser d'autres informations que le nombre de supports pour pondérer les arcs du graphe (comme les , et les résultats pratiques mettent en évidence l'intérêt de ces diverses informations pour la pondération.

3.3 Programmation mathématique à variables mixtes

Certains outils se passent de la modélisation par un graphe en concevant le problème de *scaffolding* sous la forme d'un programme mathématique à variables mixtes (ou *MIP* pour *Mixed Integer Programming*), c'est à dire que certaines variables sont sous contrainte d'entièreté (à valeurs dans $\mathbb{N}$), alors que d'autres variables peuvent prendre des valeurs dans $\mathbb{R}$.

[7] la programmation dynamique permet d'obtenir une solution optimale à un problème en considérant les solutions optimales de sous-problèmes

L'outil *MIP* ([26]) utilise ce paradigme, et bien qu'il utilise un graphe pour représenter les informations *inter-contigs*, la résolution est entièrement réalisée par programmation mathématique (en *Mixed Integer Programming*). Le graphe préliminaire permet de partitionner le graphe en sous-graphes dont il limite la taille et *MIP* les transforme en problèmes mathématiques. Ces sous-graphes sont obtenus par reconnaissance des *contigs* qui une fois supprimés partagent le graphe en deux composantes connexes ; nous les appellerons *contigs séparateurs*. Pour limiter la taille de ces composantes connexes, une construction dynamique du graphe par insertion itérative des arêtes triés en ordre décroissant en fonction de leur poids (du nombre de supports entre les deux *contigs* concernés) est initiée et lorsqu'un *contig séparateur* est repéré les deux composantes connexes sont traitées séparément. Pour limiter la taille des composantes connexes, une arête est ajoutée si et seulement si son insertion ne crée pas de composante connexe de trop grande taille. Cette limitation permet d'utiliser les arêtes les plus pondérées dans les zones denses du graphe, tout en permettant l'utilisation d'arêtes de faibles pondérations dans les zones les moins denses.

GRASS ([12]) est un autre de ces outils qui utilisent la programmation en variables mixtes pour résoudre le *scaffolding*. Une *pré-orientation* est déduite en fonction du nombre de supports que possèdent les *contigs*, de telle manière qu'un *contig* est orienté dans un sens s'il satisfait plus de supports que s'il était orienté dans l'autre, ce qui permet de réduire la complexité globale du problème. Ensuite la résolution du programme mathématique correspondant au graphe orienté est lancée. Contrairement à *MIP*, il permet de détecter les multiplicités par application d'un filtre une fois le *scaffolding* résolu.

Nous aurons présenté dans ce chapitre plusieurs méthodes de résolution du problème de *scaffolding* en essayant de proposer un panel représentatif des techniques les plus courantes. Pour autant, plusieurs autres outils connus et utilisés par la communauté de biologistes n'ont pas été cités, car leurs aspects théoriques ne semblaient pas suffisamment différents de ceux développés ici. Nous proposons au lecteur de se référer à [13] pour plus de détails et comparaisons sur les résultats pratiques de ces programmes et au tableau B.1 en Annexe B.

Chapitre 4

Problématique du stage

4.1 Motivations

Optimalité d'une solution

Nous avons pu observer, dans les chapitres 2 et 3, que les outils actuels résolvent le *scaffolding* de deux manières différentes : soit le problème est résolu de manière exacte et le passage à l'échelle n'est généralement pas possible, soit il est résolu de manière heuristique et aucune estimation de l'optimalité de la solution n'est envisageable. Pour autant, les résultats sont généralement très bons, même sur les outils qui emploient des heuristiques. Les tests d'optimalité sont effectués sur des génomes connus, ou des données artificielles (des génomes aléatoires créés par ordinateur dans le but de tester les solveurs). Dans les deux cas l'information est biaisée : dans le cas des génomes connus, leur nombre étant relativement faible, les outils ne sont pas forcément aussi efficaces sur d'autres génomes ; dans le cas des données artificielles, le biais est évidemment la randomisation informatique qui peut générer une structure dans le *graphe de scaffold* qui ne se trouve généralement pas dans la nature, ou au contraire rendre le graphe trop chaotique. Quoi qu'il en soit, dès lors que l'on séquence un génome qui n'est pas connu, soit le génome est *petit* et une résolution exacte est envisageable, soit le génome est trop *gros* et l'emploi d'un solveur heuristique est nécessaire. Dans ce dernier cas, le biologiste n'a aucune information lui permettant d'évaluer la qualité de la solution renvoyée et il ne peut qu'espérer que la solution est bonne, ou utiliser d'autres outils pour comparer les résultats, toujours sans garantie. C'est cette motivation à apporter une réponse à la question de l'optimalité qui soutient ce stage en proposant des algorithmes d'approximation polynomiaux.

Définition 11. *Un algorithme d'approximation est une heuristique qui garantie la qualité de la solution renvoyée un fournissant un ratio par rapport à la solution optimale. Ce ratio, généralement noté ρ, permet d'obtenir une borne inférieure (respectivement supérieure) à la valeur de la solution renvoyée pour un problème de maximisation (respectivement minimisation).*

Définition 12. *Un algorithme A est dit ρ-approché, avec ρ une constante, pour un problème de maximisation (respectivement de minimisation) P si pour toute instance I de P, S la valeur de la solution renvoyée par A et OPT la valeur de la solution optimale, on a $\frac{OPT}{S} \leq \rho$ (respectivement $\frac{OPT}{S} \geq \rho$).*

Un problème P qui admet un algorithme ρ-approché polynomial appartient à la classe de complexité $\mathcal{APX}$.

Prise en compte des multiplicités

Les multiplicités (de *contigs*) correspondent à la représentation multiple de mêmes *contigs* dans les données du problème. Ces *contigs* peuvent être considérés de deux manières différente : soit on considère qu'ils peuvent être contractés en un seul *contig* dont on mettra à jour les arêtes auxquelles il est adjacent, soit on les estime différents et parties de plusieurs génomes distincts, auquel cas on conserve les données intactes. Certains des outils présentés dans le chapitre 3 permettent la reconnaissance et la prise en compte des multiplicités, mais généralement les outils ne font que les éliminer sans considérer le scénario où elles doivent être conservées dans le graphe. L'intérêt de la prise en compte des multiplicités se fait ressentir lorsqu'on séquence un échantillon complexe pouvant contenir plusieurs organismes différents, par exemple le contenu d'un verre d'eau de mer. Nous voulons que notre outil soit utilisable en *métagénomique* et donc qu'il soit capable de repérer de telles multiplicités et de résoudre le problème de *scaffolding* en fonction.

4.2 Formulation

La modélisation du *graphe de scaffold* dont on connaît un couplage parfait représentant les *contigs*, formalisée dans [3] et [4], a permis à ses auteurs d'évaluer la complexité du problème de *scaffolding* sur différentes classes de graphe et de proposer deux algorithmes d'approximation à ratio constant pour le résoudre. De plus, la formalisation du problème a été redéfinie en spécifiant le nombre de *scaffolds* linéaires et circulaires que doit contenir la solution.

Dans ce chapitre, nous considérerons un graphe non-orienté $G = (V, E)$ avec un nombre pair $2n$ de sommets et sans boucle (une boucle est une arête d'un sommet vers lui même). On suppose qu'il existe un couplage parfait M^* représentant les *contigs*. Soit $\omega : E \to \mathbb{N}$ une fonction de poids sur les arêtes hors du couplage. Celles-ci représentent la manière dont les *contigs* sont liés entre eux et leurs poids indique le nombre supports, c'est à dire le nombre de paires de *reads* dont chacune des extrémités chevauche un *contig*.

Pour modéliser la structure génomique par un nombre fixé de chromosomes linéaires (chemins) et circulaires (cycles), la classe de problème étudiée est paramétrée par deux entiers, respectivement σ_p et σ_c.

Le problème de décision est formulé ainsi :

(σ_p, σ_c)-Scaffold Problem

Entrée	$G = (V, E)$ un graphe à $2n$ sommets, M^* un couplage parfait de G et $(\sigma_p, \sigma_c) \in \mathbb{N} \times \mathbb{N} \backslash \{(0, 0)\}$.
Question	Existe-t-il une collection sommets disjointe d'exactement σ_p chemins et σ_c cycles couvrant G ?

Nous nous intéresserons généralement au problème d'optimisation définit ainsi :

Max-(σ_p, σ_c)-Scaffold Problem

Entrée	$G = (V, E)$ un graphe à $2n$ sommets, M^* un couplage parfait de G et $(\sigma_p, \sigma_c) \in \mathbb{N} \times \mathbb{N} \backslash \{(0, 0)\}$.
Question	Existe-t-il une collection sommets disjointe d'exactement σ_p chemins et σ_c cycles couvrant G et dont la somme des poids des arêtes est minimale (respectivement maximale) ?

D'autres formulations du problème ont été proposées, qui contraignent encore plus le problème d'optimisation. Notamment une formulation qui définit la taille des chemins et cycles appartenant à une solution, et une formulation qui, au lieu de minimiser (ou maximiser) la somme des poids des arêtes empruntées, cherche à minimiser la longueur des cycles appartenant à la solution.

Résoudre le problème Max-(σ_p, σ_c)-Scaffold est équivalent à résoudre la formulation originelle du problème définie dans [14] qui cherche une orientation et un ordre sur les *contigs* maximisant le nombre de supports (le poids des arêtes), en considérant en plus l'existence de *scaffolds* circulaires (représentant des chromosomes circulaires) et en respectant des contraintes sur le nombre de chemins et cycles de la solution. Cette formulation peut donc être considérée comme une extension de l'originelle, et est censée représenter au mieux la structure génomique de la solution.

4.3 Résultats et problèmes ouverts

Nous présentons ci-dessous les principaux résultats théoriques des problèmes Scaffold (variantes décisionnelle et d'optimisation).

Résultats sur la complexité des problèmes Scaffold :

Problème	Complexité
(σ_p, σ_c)-Scaffold, $n = \sigma_p + 2\sigma_c$	$\mathcal{P}$
(σ_p, σ_c)-Scaffold	$\mathcal{NP}$-complet
(σ_p, σ_c)-Scaffold (bipartis planaires)	$\mathcal{NP}$-complet

Le problème de décision (σ_p, σ_c)-Scaffold vérifiant l'existence d'une solution à σ_p chemins et σ_c cycles étant NP-complet, alors sa variante d'optimisation Max/Min-(σ_p, σ_c)-Scaffold est nécessairement NP-difficile.

En plus de ces résultats sur la complexité du problème, des résultats sur les ratios d'approximation ont aussi été prouvés.

Problème	Ratio	Complexité
Max-(σ_p, σ_c)-Scaffold (complets et bipartis complets)	$\rho \leq 3$	$\mathcal{O}(n^2 \log n)$ (algorithme glouton) $\mathcal{O}(n^3)$ (couplage parfait de poids maximum)
Max-$(0, 1)$-Scaffold	$\rho \leq 2$	$\mathcal{O}(n^3)$

De la même manière, en considérant la conjecture *ETH* (*Exponential Time Hypothesis* : [15]) comme vraie, des résultats sur des bornes inférieures ont été prouvés :

Problème	Borne inférieure sur la complexité
(σ_p, σ_c)-Scaffold (bipartis)	$\mathcal{O}(2^{o(n)})$
(σ_p, σ_c)-Scaffold (bipartis planaires)	$\mathcal{O}(2^{o(\sqrt{n})})$

Définition 13. *ETH est une conjecture affirme qu'il n'existe pas d'algorithme pouvant réduire k-SAT, $k > 2$, en une complexité en temps $(O)(2^{o(n)})$.*

Ces tableaux récapitulatifs ne présentent qu'une partie des résultats théoriques de cette formulation du problème de *scaffolding*. Pour plus de résultats et leurs preuves, se référer à [3], [4], [29] et [30].

Pour résoudre le problème le plus efficacement possible (en terme de temps), il est intéressant d'étudier la complexité du problème dans des classes de graphes particulières. Quand un résultat positif est exhibé, on peut penser à la conception d'algorithmes efficaces sur ces classes de graphes. Scaffold a été étudié dans des classes de graphes particulières, en fonction de σ_p, σ_c et $\mathcal{W}_{max}$ (le poids maximal d'une arête du graphe). Nous introduisons ici quelques familles de graphes.

Définition 14. *Une quasi-forêt est un graphe $G = (V, E)$ dont on connaît un couplage parfait M^* et tel que $G \backslash M^*$ est une forêt.*

Définition 15. *Le complémentaire est graphe $H = (V, \bar{E})$ tel que H partage les mêmes sommets que G et deux sommets de H sont adjacents si et seulement si ils ne sont pas adjacents dans G.*

Définition 16. *Un graphe co-biparti est un graphe dont le complémentaire est un graphe biparti.*

On appelle *clique* un sous-ensemble de sommets d'un graphe non-orienté dont le sous-graphe induit est complet (deux sommets quelconques de la clique sont toujours adjacents), et *stable* (*ensemble indépendant* ou *independant set*) un ensemble de sommets d'un graphe deux à deux non-adjacents.

Définition 17. *Un graphe split est un graphe dont on peut partitionner les sommets en une clique et un stable.*

Ainsi, SCAFFOLD est NP-complet sur les quasi-forêts avec $\mathcal{W}_{max} = 1$, mais si on fixe $\sigma_p = 0$ alors le problème devient polynomial pour $\mathcal{W}_{max} \in \{0, 1\}$. Le problème est NP-complet sur les graphes co-bipartis et split, même pour $\mathcal{W}_{max} = 1$ (il n'existe pas d'algorithme exact en $\mathcal{O}(2^{o(m)})$) mais devient polynomial dans le cas des graphes co-bipartis avec $\mathcal{W}_{max} = 0$.

Pour autant, la complexité du problème dans les *quasi-forêts* avec $\mathcal{W}_{max} = 0$ et $\sigma_p > 0$ n'a pas encore été prouvée. Il en est de même dans les graphes split avec $\mathcal{W}_{max} = 0$. Prouver la complexité de SCAFFOLD dans ces classes de graphe sera un des objectifs de ce stage.

Parallèlement, en plus de la 3-approximation dans les graphes complets et bipartis complets, on sait qu'il n'existe pas de $n^{\frac{1}{2}-\varepsilon}$-approximation dans les quasi-forêts pour le problème MAX-(σ_p, σ_c)-SCAFFOLD. Mais aucun algorithme d'approximation dans les graphes co-bipartis ou split n'a encore été trouvé, et un objectif supplémentaire sera soit d'exposer un tel algorithme, soit de prouver qu'il ne peut en exister.

Algorithmes d'approximation

Deux algorithmes d'approximation on été développés pour résoudre le problème MAX-(σ_p, σ_c)-SCAFFOLD, avec des complexité respectives $\mathcal{O}(n^2 \log n)$ et $\mathcal{O}(n^3)$.

Le premier utilise une heuristique gloutonne qui trie les arêtes par poids décroissant et les insère itérativement. Des contraintes sous forme d'inéquation permettent d'assurer que la solution contiendra effectivement σ_p chemins et σ_c cycles.

Le deuxième utilise une heuristique se basant sur un couplage parfait de poids maximum de $E \backslash M^*$. La complexité de l'algorithme est alors dépendante de celle de l'algorithme calculant le couplage parfait de poids maximum, donc $\mathcal{O}(n^3)$.

D'autres algorithmes d'approximation peuvent éventuellement être trouvés, mais il peut aussi être intéressant d'améliorer la complexité des algorithmes actuels, comme par exemple en calculant une 2-approximation d'un couplage parfait de poids maximum en temps linéaire plutôt que cubique, et analyser les résultats pratiques ensuite.

Des résultats sur des données artificielles et réelles ont été faits et sont décris dans [30]. Ces résultats montrent qu'en plus d'obtenir une borne inférieure théorique, les solutions sont en pratique très proches de l'optimal. L'algorithme utilisant le couplage parfait de poids maximum semble plus performant en pratique que l'algorithme glouton, sauf sur les graphes très peu denses.

Mesure différentielle

Une utilisation particulière des algorithmes d'approximation peut être envisagée. En effet, comme nous connaissons un ratio à l'optimal dans le pire des cas, on peut essayer de générer une *mauvaise* solution qui conserve ce ratio pour évaluer d'autres solutions générées par des heuristiques (ou métaheuristiques) qui ne propose *a priori* aucune garantie quant à la qualité de sa solution. Cette mesure est directement inspirée des résultats de [7], repris ensuite dans [22].

Considérons $\mathcal{A}$, un algorithme ρ-approché pour le problème Max-(σ_p, σ_c)-Scaffold ($\rho \in [1, +\infty[$), et $\mathcal{H}$ une heuristique sans garantie de performance. Pour toute instance I de Max-(σ_p, σ_c)-Scaffold, on note $S_I^{\mathcal{A}}$ et $S_I^{\mathcal{H}}$ respectivement la solution retournée par $\mathcal{A}$ sur l'instance I et la solution retournée par H sur l'instance I. Soient OPT_I la valeur de la solution optimale de l'instance I, $\omega_I^{\mathcal{A}}$ et $\omega_I^{\mathcal{H}}$ les valeurs respectives des solutions $S_I^{\mathcal{A}}$ et $S_I^{\mathcal{H}}$. On sait par définition de $\mathcal{A}$ que $\omega_I^{\mathcal{A}} \geq \frac{OPT_I}{\rho}$, donc $\rho \times \omega_I^{\mathcal{A}} \geq OPT_I$. On peut définir $P_{\mathcal{H}} = \frac{\omega_I^{\mathcal{H}}}{OPT_I}$ comme le rapport à l'optimal de la solution $\mathcal{H}_I$. Comme OPT_I est inconnue, on peut considérer l'inégalité $P_{\mathcal{H}} \geq \frac{\omega_I^{\mathcal{H}}}{\rho \times \omega_I^{\mathcal{A}}}$, ce qui nous donne une borne inférieure quant à l'optimalité de la solution retournée par l'heuristique $\mathcal{H}$.

Notons que, dans certains cas, la *mauvaise* solution espérée que retourne l'algorithme d'approximation peut en fait être *bonne*, ce qui augmente la borne supérieure sur la valeur d'une solution optimale. Dans ce cas, même des bornes supérieures triviales peuvent améliorer la mesure.

Chapitre 5

Poursuite du stage

Dans le chapitre 4 nous avons commencé à énoncer certains des objectifs de ce stage. Notamment, la prise en compte des multiplicités dans le modèle existant, l'utilisation de nouvelles heuristiques pour la conception d'algorithmes d'approximations, la classification du problème de Scaffold dans certaines classes de graphes et l'évaluation de l'optimalité d'une solution par mesure différentielle.

Définition 18. *Un noyau, ou kernel, est une réduction de l'instance d'origine, possiblement plus petite, d'un problème paramétré par k dont la taille est bornée par une fonction qui ne dépend que du paramètre k.*

On peut ajouter à cette liste la recherche d'un noyau polynomial ou exponentiel permettant de mettre au point des algorithmes exacts passant mieux à l'échelle, ou prouver leur non-existence, et la recherche de nouvelles formulations ou modélisations qui pourraient renforcer notre connaissance théorique sur le problème de Scaffold. Nous proposons en Annexe C une formalisation des problèmes ouverts que nous voulons aborder durant le stage.

Parmi les paramètres intéressant sur lesquels nous pourrions chercher un noyau, nous citerons bien entendu σ_p et σ_c, mais éventuellement aussi des paramètres plus structurels du graphe comme la *treewidth* ou la *branchwidth*...

À la suite de ce stage, nous voulons proposer plusieurs solutions théoriques au problème de *scaffolding* avec prise en compte des multiplicités, en particulier quand il est contraint sur le nombre de chemins et cycles. En parallèle, le développement de l'outil utilisant les algorithmes d'approximation sera poursuivi, ce qui nous permettra d'évaluer les résultats pratiques et de le comparer aux outils existants en terme de temps d'exécution et de qualité de solution.

Cette modélisation, plus contrainte que le modélisation originelle du problème de *scaffolding*, et sa formulation via un couplage parfait ont permis d'apporter de nombreux résultats théoriques et de réelles avancées sur la qualité des solutions. L'utilisation d'algorithmes d'approximation à garantie de performance est une approche novatrice et riche de sens dans ce contexte et permet d'étudier le problème de *scaffolding* sous un nouvel angle. Poursuivre l'analyse théorique pour améliorer les résultats pratiques, telle est la devise de ce projet.

Annexe A

Figure

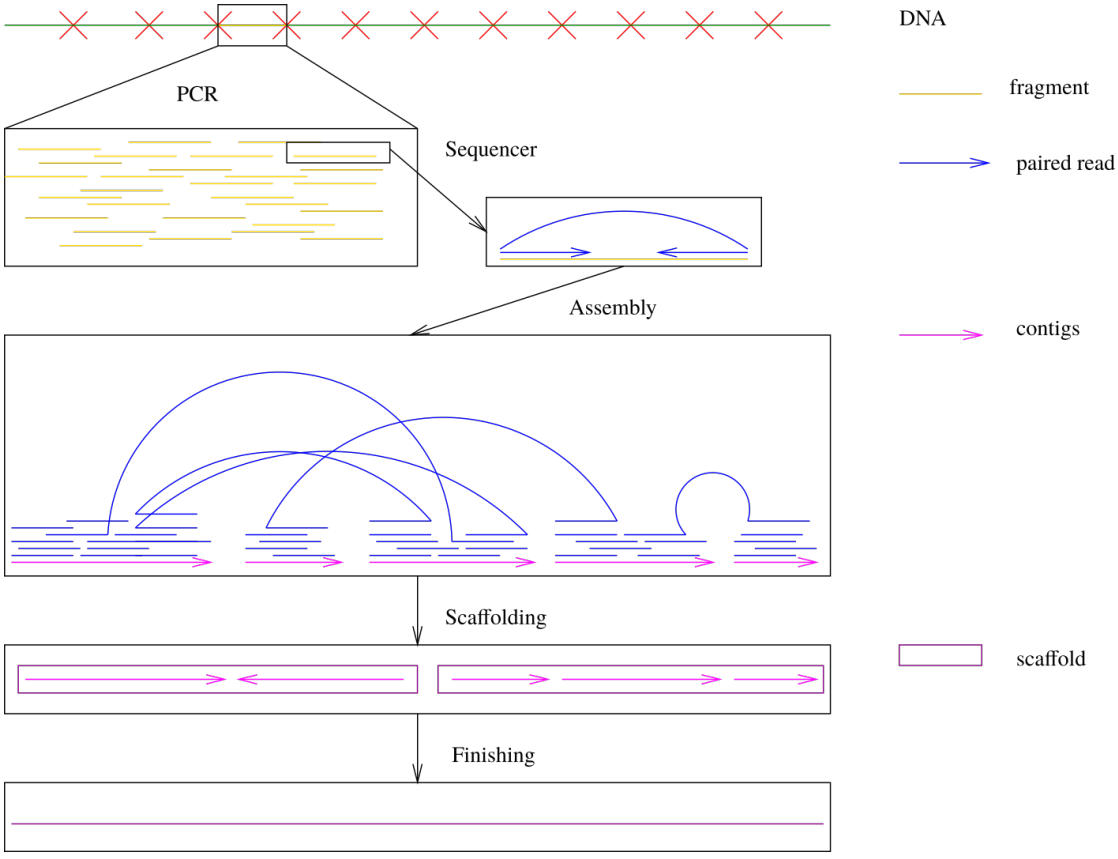


Figure A.1 – Illustration des différentes étapes du processus de séquençage

Annexe **B**

Tableau comparatif

Outil	Structure de données	Technique de résolution	Prise en compte des multiplicités	Nombre de citations (approximatif)
Bambus	Graphe	Glouton	Non	182
SSPACE	Table de hachage	Glouton	Non	572
SSAKE	Table de hachage	Glouton	Non	442
Bambus 2	Graphe	Reconnaissance de structures	Oui	77
SCARPA	Graphe	FPT	Non	37
OPERA	Graphe	Contraction	Non	115
BESST	Graphe	Analyse statistique	Non	14
SOPRA	Graphe	Analyse statistique	Non	98
MIP	Programme mathématique & Graphe	Mixed Integer Programming	Non	50
GRASS	Programme mathématique & Graphe	Mixed Integer Programming	Oui	28

Table B.1 – Tableau comparatif des outils du Chapitre 3

Annexe C

Problèmes ouverts

Entrée	$G = (V, E)$ un graphe à $2n$ sommets, M^* un couplage parfait de G, $\sigma_p \in \mathbb{N}^*$, $\sigma_c \in \mathbb{N}$, $\mathcal{W}_{max} = 0$ et tel que $G \backslash M^*$ est une forêt.
Question	Résoudre Scaffold sur G est-il NP-difficile ?

Entrée	$G = (V, E)$ un graphe à $2n$ sommets, M^* un couplage parfait de G, $\sigma_p \in \mathbb{N}$, $\sigma_c \in \mathbb{N}$, $\mathcal{W}_{max} = 0$ et tel que G est un graphe split.
Question	Résoudre Scaffold sur G est-il NP-difficile ?

Entrée	$G = (V, E)$ un graphe à $2n$ sommets, M^* un couplage parfait de G, $\sigma_p \in \mathbb{N}$, $\sigma_c \in \mathbb{N}$ et tel que G est un graphe co-biparti ou un graphe split.
Question	Résoudre Max-(σ_p, σ_c)-Scaffold sur G est-il NP-difficile ? Si oui, peut-on trouver des algorithmes d'approximation à facteur contant ?

Entrée	$G = (V, E)$ un graphe à $2n$ sommets, M^* un couplage parfait de G, $\sigma_p \in \mathbb{N}$, $\sigma_c \in \mathbb{N}$.
Question	Existe-t-il un paramètre k de G qui permet d'obtenir un noyau pour le problème Scaffold ?

Bibliographie

[1] M. Boetzer, C. Henkel, H. Jansen, D. Butler, and W. Pirovano. Scaffolding preassembled contigs using SSPACE. *Bioinformatics*, 27(4) :578–579, 2011.

[2] E. Castillo, A. Gonejo, P. Pedregal, R. Garciá, and N. Alguacil. *Nonlinear Programming*, pages 47–70. John Wiley & Sons, Inc., 2001.

[3] A. Chateau and R. Giroudeau. Complexity and polynomial-time approximation algorithms around the scaffolding problem. In *Algorithms for Computational Biology - Spain, July 1-3, 2014*, pages 47–58, 2014.

[4] A. Chateau and R. Giroudeau. A complexity and approximation framework for the maximization scaffolding problem. *Theoretical Computer Science*, 595 :92–106, 2015.

[5] Chauve, Patterson, and Rajaraman. Hypergraph covering problems motivated by genome assembly questions. In *Combinatorial Algorithms*, pages 428–432. Springer, 2013.

[6] A. Dayarian, T. Michael, and A. Sengupta. SOPRA : Scaffolding algorithm for paired reads via statistical optimization. *BMC bioinformatics*, 11(1) :1, 2010.

[7] M. Demange and V. Th. Paschos. On an approximation measure founded on the links between optimization and polynomial approximation theory. *Theoretical Computer Science*, 158 :117–141, 1996.

[8] N. Donmez and M. Brudno. SCARPA : scaffolding reads with practical algorithms. *Bioinformatics*, 29(4) :428–434, 2013.

[9] P. Eades, X. Lin, and W. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6) :319–323, 1993.

[10] S. Gao, W.-K. Sung, and N. Nagarajan. Opera : Reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *Journal of Computational Biology*, 18(11) :1681–1691, 2011.

[11] M. Garey and D. Johnson. Computers and intractability. *A Guide to the Theory of NP-Completeness*, 1979.

[12] A. Gritsenko, J. Nijkamp, M. Reinders, and D. de Ridder. Grass : a generic algorithm for scaffolding next-generation sequencing assemblies. *Bioinformatics*, 28(11) :1429–1437, 2012.

[13] M. Hunt, C. Newbold, M. Berriman, and T. Otto. A comprehensive evaluation of assembly scaffolding tools. *Genome Biology*, 15(3) :1–15, 2014.

[14] D. H. Huson, K. Reinert, and E. W. Myers. The greedy path-merging algorithm for contig scaffolding. *J. ACM*, 49(5) :603–615, 2002.

[15] R. Impagliazzo and R. Paturi. The complexity of k-sat. *2012 IEEE 27th Conference on Computational Complexity*, 0 :237, 1999.

[16] R. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[17] J. Kececioglu and E Myers. Combinatorial algorithms for DNA sequence assembly. *Algorithmica*, 13(1-2) :7–51, 1995.

[18] S. Koren, T. Treangen, and M. Pop. Bambus 2 : scaffolding metagenomes. *Bioinformatics*, 27(21) :2964–2971, 2011.

[19] B. Langmead, C. Trapnell, M. Pop, S. Salzberg, et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biol*, 10(3) :R25, 2009.

[20] D. Lokshtanov, S. Saurabh, and S. Sikdar. Simpler parameterized algorithm for OCT. In *Combinatorial algorithms*, pages 380–384. Springer, 2009.

[21] J. Meidanis, O. Porto, and G. P. Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88(1-3) :325–354, 1998.

[22] V. Paschos. *Complexité et approximation polynomiale*. Hermes Science Publications, 2004.

[23] M. Pop, D. Kosack, and S. Salzberg. Hierarchical scaffolding with Bambus. *Genome research*, 14(1) :149–159, 2004.

[24] J. A. Reuter, D. V. Spacek, and M. P. Snyder. High-throughput sequencing technologies. *Molecular Cell*, 58(4) :586 – 597, 2015.

[25] K. Sahlin, F. Vezzi, B. Nystedt, J. Lundeberg, and L. Arvestad. BESST-efficient scaffolding of large fragmented assemblies. *BMC bioinformatics*, 15(1) :1, 2014.

[26] L. Salmela, V. Mäkinen, N. Välimäki, J. Ylinen, and E. Ukkonen. Fast scaffolding with small independent mixed integer programs. *Bioinformatics*, 27(23) :3259–3265, 2011.

[27] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.

[28] R. Warren, G. Sutton, S. Jones, and R. Holt. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4) :500–501, 2007.

[29] M. Weller, A. Chateau, and R. Giroudeau. Exact approaches for scaffolding. *BMC bioinformatics*, 16(Suppl 14) :S2, 2015.

[30] M. Weller, A. Chateau, and R. Giroudeau. On the implementation of polynomial-time approximation algorithms for scaffold problems. *En soumission*, 2015.

Glossaire

Algorithme ρ-approché

Un algorithme A est dit ρ-approché, avec ρ une constante, pour un problème de maximisation (respectivement de minimisation) P si pour toute instance I de P, S la valeur de la solution renvoyée par A et OPT la valeur de la solution optimale, on a $\frac{OPT}{S} \leq \rho$ (respectivement $\frac{OPT}{S} \geq \rho$).

Algorithme d'approximation

Heuristique qui garantie la qualité de la solution renvoyée un fournissant un ratio par rapport à la solution optimale. Ce ratio, généralement noté ρ, permet d'obtenir une borne inférieure (respectivement supérieure) à la valeur de la solution renvoyée pour un problème de maximisation (respectivement minimisation).

Algorithme de complexité FPT

Algorithme qui permet de résoudre un problème paramétré par k en un temps $f(k) \cdot poly(n)$ où f est une fonction quelconque et $poly$ une fonction polynomiale.

Algorithme *glouton*

Algorithme qui n'envisage qu'un seul choix améliorant sa solution actuelle à chaque étape.

Biparti

Graphe dont il existe une partition de son ensemble de sommets en deux sous-ensemble $\mathcal{U}$ et $\mathcal{V}$ telle que chaque arête possède une extrémité dans $\mathcal{U}$ et l'autre dans $\mathcal{V}$.

Clique

Sous-ensemble de sommets d'un graphe non-orienté dont le sous-graphe induit est complet.

Complémentaire d'un graphe $G = (V, E)$

Graphe $H = (V, \bar{E})$ tel que H partage les mêmes sommets que G et deux sommets de H sont adjacents si et seulement si ils ne sont pas adjacents dans G.

Contigs

Fragments de génome qu'il faut orienter et ordonner de façon à reconstituer le génome entier. Ces fragments sont des séquences d'`ADN` dont on connaît la longueur et le code génétique les constituant. Plus formellement, ce sont des mots de longueur variable sur l'alphabet $\mathcal{A}^*$ avec $\mathcal{A} = \{A, C, G, T\}$.

Couplage

Ensemble d'arête d'un graphe tel qu'aucune paire d'arêtes de ce couplage ne partage de sommets.

Couplage parfait

Ensemble d'arête d'un graphe formant un *couplage* tel que tout sommet du graphe est incident à exactement une arête du couplage.

ETH

Conjecture affirme qu'il n'existe pas d'algorithme pouvant réduire k-SAT, $k > 2$, en une complexité en temps $(O)(2^{o(n)})$.

Graphe co-biparti

Graphe dont le *complémentaire* est un graphe *biparti*.

Graphe *split*

Graphe dont on peut partitionner les sommets en une *clique* et un *stable*.

K-mer

Sous-chaîne de taille k dans une chaîne.

Mapping

Opération d'alignement des *reads* sur les *contigs*. De cette opération, on déduit les arêtes *inter-contigs* (ou arcs suivant le type de graphe) et leurs poids en fonction du nombre de supports que partagent les *contigs* en extrémité.

Noyau, ou kernel

Réduction de l'instance d'origine, possiblement plus petite, d'un problème paramétré par k dont la taille est bornée par une fonction qui ne dépend que du paramètre k.

Programme mathématique à variables mixtes

Problème sous forme d'équations et d'inéquations sur des variables entières ou réelles (donc mixtes) où l'on cherche à minimiser, ou maximiser, une fonction (appelée fonction objectif) tout en satisfaisant les contraintes imposées par les inéquations.

Quasi-forêt

Un graphe $G = (V, E)$ dont on connaît un *couplage parfait* M^* et tel que $G \backslash M^*$ est une forêt.

Read

Extrémité d'un fragment d'`ADN` dont la séquence génomique est connue.

Scaffold

Ensemble de *contigs* orientés et consécutifs formant une séquence génomique linéaire ou cyclique.

Stable

Ensemble de sommets d'un graphe deux à deux non-adjacents.

Support

Paire de *reads* qui possède une extrémité chevauchant un *contig* et l'autre extrémité chevauchant un deuxième *contig*.