Détection de régions génomiques homologues par un algorithme de flots avec coûts

Eric Audemard, Thomas Faraut, Thomas Schiex

Détection de régions génomiques homologues par un algorithme de flot avec coûts

Eric Audemard, Thomas Faraut, Thomas Schiex

INRA toulouse ; Chemin de Borde Rouge BP 52627, 31326 Castanet Tolosan cedex , France
{eric.audemard,thomas.faraut,thomas.schiex}@toulouse.inra.fr

Résumé : *L'identification de régions génomiques homologues, c'est à dire possédant une origine ancestrale commune, est centrale à l'étude des génomes. La difficulté résulte de l'érosion des traces de cette relation d'homologie. Le principe est alors de rechercher des signatures d'homologie parmi un ensemble de courtes régions similaires. Nous présentons un formalisme modélisant toutes les solutions dans un graphe. Dans ce graphe, un chemin est une représentation de deux régions potentiellement homologues et notre but est de retrouver un ensemble cohérent de chemins, c'est à dire de régions homologues. Cet ensemble est reconstruit à l'aide d'une méthode d'optimisation globale basée sur la théorie des flots.*

Mots-Clés : *bioinformatique, évolution des génomes, génomique comparative, flots, réseaux de transport.*

1 Introduction

La recherche d'une relation de parenté, c'est à dire d'une origine ancestrale commune, appelée également relation d'homologie, entre gènes d'espèces différentes, ou au sein d'une même espèce, est une étape importante de l'étude des génomes. La mise en évidence d'une origine ancestrale commune de deux gènes, ou plus généralement de deux régions porteuses d'une fonction biologique, fournit des arguments en faveur d'une fonction biologique commune. Cette recherche d'origine ancestrale commune est surtout centrale à l'étude de l'évolution des gènes, des génomes et des espèces. Avec l'arrivée des génomes entièrement séquencés, cette recherche de relations de parenté se transpose naturellement aux segments des chromosomes. L'origine commune des espèces implique que deux espèces quelconques partageaient la même organisation chromosomique chez leur plus proche parent. Pour ces deux espèces, les organisations chromosomiques actuelles sont le résultat d'une histoire évolutive propre, puisque indépendante depuis leur séparation (spéciation), faite de mutations chromosomiques de différentes natures (de la substitution de nucléotides à des réarrangements chromosomiques pouvant impliquer l'échange de grands fragments entre chromosomes). Une forte similitude entre deux régions chromosomiques suggère une conservation depuis leur plus proche parent. On parle alors de segments chromosomiques conservés, ou de segments homologues, et l'identification de cette conservation correspond à la mise en évidence d'une origine ancestrale commune pour ces segments chromosomiques et permet d'appliquer, à cette échelle, la démarche d'inférence de fonction et l'étude de l'évolution des génomes. La recherche de segments homologues au sein de génomes entièrement séquencés fait l'objet du présent travail.

Peu après la séparation de deux espèces, la conservation des génomes est telle qu'il est facile de les comparer (aligner) et d'identifier les longs segments chromosomiques conservés. Lorsque la spéciation

est plus ancienne, l'identification des régions conservées est rendue difficile par la poursuite des différents mécanismes évolutifs (mutations ponctuelles, réarrangements...). Ces modifications peuvent mener à la disparition locale de la similarité, à des changement d'orientation ou d'ordre entre éléments inclus dans la région (gènes, régulateurs...). Les seuls indices aisément identifiables permettant de retrouver ces régions homologues sont de courtes régions suffisamment similaires pour pouvoir être alignées (aussi appelées ancres), traces potentielles de l'homologie entre les deux régions. La densité de ces ancres dans une région, la succession d'une quantité importante d'ancres dans un ordre et/ou une orientation identique sont autant d'éléments permettant de suggérer l'existence d'une région homologue.

L'identification de segments homologues entre deux génomes est une extension naturelle du problème d'alignement entre deux séquences. L'une des formalisations de ce problème d'alignement identifie l'alignement à un chemin dans un graphe. Nous proposons de formaliser le problème de la recherche de segments homologues sous la forme d'une recherche d'un ensemble de chemins dans un graphe. Cette formalisation n'est pas nouvelle, elle a été utilisée implicitement ou explicitement par de nombreuses méthodes [9, 3, 8, 14, 15] dédiés à la détection de segments homologues. Une fois le graphe créée, elles utilisent une approche gloutonne, utilisant un algorithme de "plus court chemin", pour sélectionner les chemins un par un. D'autres méthodes [16, 12, 10, 13] de type *cluster* sont aussi développées, mais la sélection des chaînes reste gloutonne. La principale nouveauté de notre approche réside dans le choix de reconstruire simultanément un ensemble cohérent de chemins et dans l'utilisation d'un algorithme de "flot de coût mininimum" pour ce faire. Nous montrons également comment la prise en compte de contraintes supplémentaires permet d'adapter simplement l'algorithme à l'identification de segments homologues au sein d'un même génome.

L'article est organisé de la manière suivante. Nous commençons par rappeler brièvement le problème biologique et sa formalisation sous forme de la recherche de chemins dans un graphe. Nous montrons ensuite comment une adaptation de l'algorithme de "flot maximum coût mininimum" permet de résoudre le problème. Nous présenterons enfin une étude comparative des résultats de notre algorithme avec une approche gloutonne d'une part et avec des logiciels publiés et dédiés à ce problème d'autre part.

2 Modélisation du problème

2.1 Le problème biologique

L'évolution des chromosomes est donc un processus qui se déroule à différentes échelles. Aux modifications locales n'impliquant que quelques nucléotides, s'ajoutent des réarrangements microscopiques locaux pouvant impliquer de quelques milliers à quelques dizaines de milliers de nucléotides et des réarrangements de plus grande ampleur portant sur plusieurs millions de nucléotides voire des bras chromosomiques entiers. Ces deux types de réarrangements, microscopiques et macroscopiques, n'ont pas la même dynamique, les seconds étant bien moins fréquent que les premiers (un génome de mammifère subit quelques dizaines de mutations par an et à peine un réarrangement de grande ampleur par million d'années). Pour illustrer l'évolution des génomes, on peut utiliser l'analogie d'un long texte organisé en chapitres. Les modifications microscopiques locales fréquentes porteraient sur l'orthographe des mots, leur substitution ou un réarrangement local du texte alors que les réarrangements macroscopiques déplaceraient de grandes sections de texte au sein d'un chapitre ou même entre chapitres. Notre problème consiste, à partir de l'observation de deux livres ayant évolués indépendamment, à identifier de longues portions de texte globalement conservées. La difficulté réside dans le fait que les modifications microscopiques locales brouillent le signal évolutif de conservation.

Le principe de détection de segments homologues consiste à exploiter des similitudes locales fortes, identifiées au préalable. Ces similitudes définissent des points d'ancrage sur chacun des deux génomes A et B qui seront exploitées pour identifier les segments homologues. Dans la suite du texte, ces similitudes qui établissent une correspondance locale entre les deux génomes seront appelés ancres. Comme l'illustre la figure 1, l'organisation interne de ces similitudes, au sein des segments homologues, est susceptible d'être modifiée au cours de l'évolution : modification de l'ordre ou de l'orientation[1] La méthode de reconstruction exploitera le degré de conservation de l'organisation de ces ancres pour identifier les segments homologues.

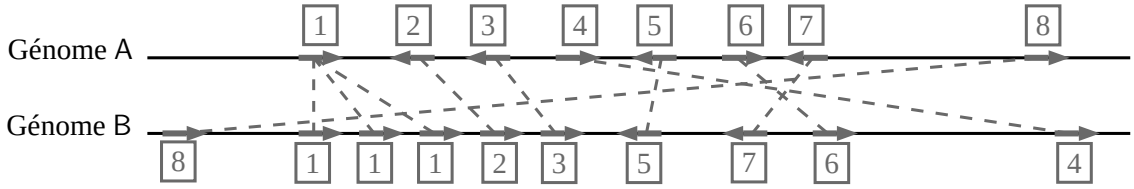

FIG. 1 – Exemple de deux régions homologues entre le génome A et le génome B qui ont subit des évènements évolutifs au cours du temps. Une ancre est représentée par deux flèches de même numéro sur chacun des génomes. Le sens de la flèche représente le brin d'ADN. Les ancres 6 et 7 sont des ancres de polarités négatives. Les ancres 4 et 5 sont des ancres de polarités positives. L'enchaînement des ancres 5, 6, 7 sur le génome A et 5, 7, 6 sur le génome B ; montre un changement d'ordre. Les ancres 3 et 5 se sont pas la même distance sur chacun des génomes.

2.2 Formalisme

Nous considérons dans la suite que les génomes étudiés sont constitués chacun d'un seul chromosome, la méthode se généralisant facilement au cas multi-chromosomique. Les génomes A et B sont donc modélisés par deux chaînes de caractères, de tailles respectives n et m, sur un alphabet de quatre lettres $\{A, C, G, T\}$. Un segment d'un génome est entièrement déterminé par la donnée de l'index de la première lettre et de celui de la dernière lettre, il peut donc être noté par un intervalle $I = [deb, fin]$, avec $1 \leq deb < fin$ ($fin \leq n$ pour un intervalle de A et $fin \leq m$ pour un intervalle de B). On munit ces intervalles d'une relation d'ordre, on note $I < I'$ le fait que l'intervalle I' soit situé strictement après l'intervalle I. La distance entre 2 intervalles, définie comme la distance séparant ses éléments les plus proches, sera notée $dist(I, I')$.

Nous allons définir un graphe dont les sommets sont constitués des ancres et nous ramènerons le problème à la recherche de chemins dans ce graphe.

2.3 Les ancres comme sommets du graphe

La première étape de l'identification de segments homologues entre deux génomes A et B se fait par la recherche d'*ancres*, ou paires de régions génomiques dont la similarité est significative et qui s'appuie sur l'utilisation d'outils d'alignement locaux tels que Blast [1]. Dans tous les cas, l'entrée du processus de détection de segments homologues est formée d'un ensemble $\mathcal{A}$ d'ancres, une notion que nous précisons maintenant :

Une ancre a définit une correspondance entre un segment de A et un segment de B. On note a^{A} et a^{B} les intervalles correspondants. Une ancre est susceptible de relier des éléments d'orientations

[1]Il est important de noter que les éléments génétiques portés par l'ADN sont orientés, c'est en particulier le cas des gènes. L'orientation est définie par le sens de lecture de l'information génétique.

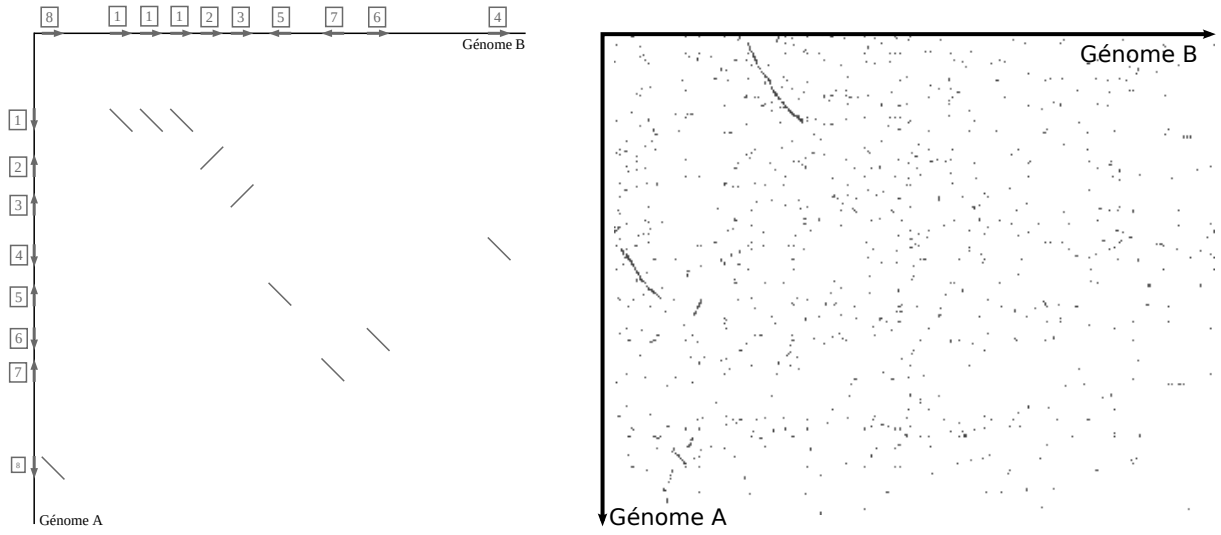

FIG. 2 – A gauche le dotplot théorique correspondant à la comparaison du génome A et B de la figure 1. A droite, une partie d'un dotplot d'une comparaison du génome d'*Arabidopsis thaliana* contre lui-même. Des régions homologues, formées d'un grand nombre d'ancres, apparaissent sous forme de diagonales sombres.

opposées, on dira alors qu'elle est de polarité négative, elle est de polarité positive sinon. Enfin, la qualité de la similarité observée entre les deux régions est capturée dans un score (fourni par les outils d'alignement locaux). Une ancre a_i est donc finalement déterminée par la donnée de quatre composantes : un intervalle a_i^{A} sur le génome A, un intervalle a_i^{B} sur le génome B, une polarité notée $a_i.pol$ et un score noté $a_i.s$.

On appelle *dotplot* la représentation graphique en deux dimensions de ces ancres (voir figure 2). Les deux axes du dotplot représentent les deux génomes, les intervalles $[1, n]$ et $[1, m]$, et chaque ancre a_i est représentée par un segment de droite dans cette représentation en deux dimensions dont les projections sur les deux axes correspondent exactement aux intervalles a_i^{A} et a_i^{B}. La pente de ce segment de droite est négative si a_i est de polarité positive, $a_i.pol = +$ (figure 2, ancres 6 et 7), et positive si a_i est de polarité négative, $a_i.pol = -$ (figure 2, ancres 2 et 3).

2.4 Le voisinage et la compatibilité des ancres

Un certain nombre de propriétés permettent de distinguer quand il est raisonnablement possible de supposer que deux ancres $a_i, a_j \in \mathcal{A}$ font partie d'un même segment ancestral.

Une première condition naturelle est d'exiger que les deux ancres ne soient pas trop éloignées physiquement l'une de l'autre sur les deux génomes, à une distance bornée notée $distMax^{\mathsf{g}}$ (dépendant du génome considéré).

$$dist(a_i^{\mathsf{g}}, a_j^{\mathsf{g}}) < distMax^{\mathsf{g}} \tag{1}$$

Définition 1 (Ordre sur les ancres) *Soit deux ancres $a_i, a_j \in \mathcal{A}^2$. On dit que $a_i \prec a_j$ lorsque a_i et a_j respectent la condition 1 et que $a_i^{\mathsf{A}} < a_j^{\mathsf{A}}$, c'est à dire a_i précède a_j sur le génome A.*

On peut noter que la relation $\prec$ définit un ordre strict partiel sur les ancres. Cet ordre partiel

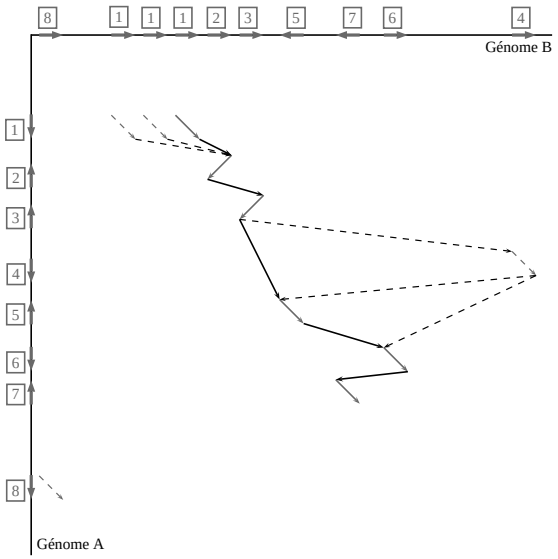

FIG. 3 – Graphe de relation R construit à partir du dotplot à gauche de la figure 2. Les arcs et sommets non utilisés dans une chaîne sont en pointillés.

strict se capture naturellement sous la forme d'un graphe orienté, appelé *graphe de relation* :

Définition 2 (Graphe de relation) *Pour un ensemble d'ancres donné A, le graphe de relation R associé à $\mathcal{A}$ est un graphe orienté dont l'ensemble des sommets est $\mathcal{A}$ et qui contient l'arc (a_i, a_j) si et seulement si $a_i \prec a_j$.*

Les arcs de ce graphe seront de plus pondérés et nous préciserons ces pondérations au paragraphe 2.8.

La relation $\prec$ étant un ordre, ce graphe est sans circuit. Tout chemin dans ce graphe représente une succession d'ancres qui respectent la condition 1 et qui sera appelé une chaîne par la suite. Toute chaîne définit donc un paire de segments homologues potentielle. Afin de différencier les chaînes, un score sera associé à chaque arc et sommet de R avec l'idée que plus le score est élevé, plus il est vraisemblable que l'élément appartienne à une paire de segments homologues.

Des contraintes supplémentaires peuvent être imposées pour les ancres qui participent à un segment ancestral : une contrainte de polarité (toutes les ancres doivent avoir la même polarité) ou une contrainte sur la conservation de l'ordre des ancres sur les deux génomes (l'ordre doit être conservé si la polarité est positive et totalement inversé le cas contraire). Un certain nombre d'outils existant imposent ces contraintes supplémentaires. Nous ne les utiliserons pas mais l'ensemble de ce travail s'applique de façon immédiate si l'on souhaite exploiter ces propriétés supplémentaires et l'implémentation permet d'imposer ces conditions supplémentaires.

2.5 Les chaînes et les ancres

Une chaîne est un chemin dans le graphe de relation R. Par définition de la relation $\prec$, toute chaîne c traverse des ancres dont les intervalles se succèdent sur le génome A (mais pas nécessairement sur le génome B). A chaque chaîne c, et pour chaque génome $\mathsf{g} \in \{A, B\}$, on peut associer un ensemble d'informations : le début de la chaîne $c^{\mathsf{g}}.deb$, défini par la plus petite position d'une ancre dans c sur g, et sa fin $c^{\mathsf{g}}.fin$ (définie à partir de la plus grande position, on a donc $c^{\mathsf{g}}.deb < c^{\mathsf{g}}.fin_x$). Comme

pour les ancres, une chaîne c définit donc deux intervalles sur chacun des génomes, notés c^{A} et c^{B}. Le nombre d'ancres dans la chaîne est noté $c.n$.

À partir de l'ensemble des positions de début et de fin de l'ensemble des ancres traversées par une chaîne, il est possible de calculer un coefficient de corrélation de Bravais-Pearson entre les coordonnées sur le génome A et sur le génome B. Ce coefficient, noté $c.corr$, renseigne sur le degré de dépendance linéaire entre les coordonnées et fournit une mesure de la conservation de l'organisation des ancres du segment sur les deux génomes. Le signe de ce coefficient définit la polarité $c.pol$ d'une chaîne c. Enfin, le score d'une chaîne $c.s$ est défini comme la somme des scores des éléments (sommets et arcs) qui la composent.

Tout comme pour les ancres, il est possible de définir un ensemble de propriétés destinées à capturer les chaînes qui sont les plus représentatives de paires de segments homologues. Ces conditions sont appelées par la suite conditions de *validité*. Une première série de conditions vise à éliminer les chaînes trop petites (contenant peu d'ancres) ou de trop mauvaises qualité (de score faible ou avec une coefficient de corrélation trop proche de 0) :

Soit $minAncre$ le nombre minimum d'ancres d'une chaîne, $minScore$ le score minimum d'une chaîne et $minCorr$ le coefficient de corrélation minimum d'une chaîne, une chaîne valide doit respecter les conditions suivantes :

$$\begin{array}{rcl} c_i.s & \geq & minScore \\ c_i.n & \geq & minAncre \\ c_i.corr & \geq & minCorr \end{array} \tag{2}$$

Le problème de recherche de segments homologues entre deux génomes pourrait alors se ramener à la production d'un ensemble $\mathcal{C}$ de chaînes tirées de R, qui respectent les conditions précédentes et dont la somme des scores est maximum. Il est toutefois nécessaire de rajouter un ensemble de conditions liant entre elles les différentes chaînes apparaissant dans l'ensemble de chaîne $\mathcal{C}$.

Ainsi, une ancre ne peut pas appartenir à deux chaînes distinctes. Chaque ancre est en effet l'image d'un unique évènement de duplication (une ancre implique deux régions similaires seulement) et ne peut pas servir à en expliquer plusieurs.

$$\forall c, c' \in \mathcal{C}, \text{Si } a \in c \quad \text{alors } a \notin c' \tag{3}$$

De même, un couple de chaînes c_i, c_j de $\mathcal{C}$ connectées dans R ne peuvent pas se chevaucher à la fois sur le génome A et sur le génome B. La duplication d'une région implique en effet la création d'une région qui n'est l'image que d'une unique origine.

$$(c_i^{\mathsf{A}} \cap c_j^{\mathsf{A}}) = \varnothing \quad \text{ou} \quad (c_i^{\mathsf{B}} \cap c_j^{\mathsf{B}}) = \varnothing \tag{4}$$

Ces deux conditions supplémentaires montrent que l'ensemble de chaînes qu'il faut produire n'est pas un ensemble de chaînes indépendantes mais qu'il forme un tout.

2.6 Le cas des duplications segmentales

Dans le cas particulier de la détection des duplications segmentales (copie de segments chromosomiques à l'intérieur d'un même génome), des contraintes supplémentaires découlent du fait que les deux génomes comparés sont identiques. Cela impose qu'une région du génome ne peut pas participer à une chaîne à la fois en tant que séquence du génome A et en tant que séquence du génome B.

La condition 4 précédente se spécialise. Une chaîne c qui met en correspondance des régions d'un même génome, ne peut pas se chevaucher elle-même car le résultat d'une duplication ne peut chevaucher son origine.

$$c_i^{\mathsf{A}} \cap c_i^{\mathsf{B}} = \varnothing \tag{5}$$

Cette condition empêche de fait la construction d'une chaîne unique qui correspondrait à une région dupliquée trois fois ou plus consécutivement (phénomène classique et appelé *duplications en tandem*). Plusieurs chaînes seront alors nécessaires.

Cette condition au niveau des chaînes permet de construire une condition nécessaire au niveau des ancres : deux ancres qui mettent en relation des régions ne peuvent pas appartenir à une même chaîne si la simple chaîne c formée de ces deux ancres définit deux régions qui se chevauchent.

Soit a_i et a_j deux ancres satisfaisant $a_i \prec a_j$ et soit c la chaîne formée des deux ancres a_i et a_j. Alors :

$$c^{\mathsf{A}} \cap c^{\mathsf{B}} = \varnothing. \tag{6}$$

Dans le cas d'analyse de duplications segmentales, cette condition sera donc directement mobilisée en supprimant dans R tout arc reliant deux ancres a_i et a_j violant cette propriété. Cette condition est nécessaire mais non suffisante.

2.7 Problème d'optimisation

Une fois le graphe R de relations construit, le problème de recherche de régions homologues entre A et B se ramène donc à la recherche d'un ensemble de chemins de score global maximum et vérifiant un ensemble de propriétés (propriétés 1, 3, 2 et 4 au moins). La taille de cet ensemble est inconnue a priori.

La contrainte 1, qui exige une distance maximale entre ancres successives, est immédiate à prendre en compte dans le graphe de relation R et est donc satisfaite dans la majorité des outils existants, comme DAGchainer [8]. Les autres contraintes sont plus délicates à traiter. L'approche suivie par DAGchainer consiste à résoudre le problème d'un ensemble de chaîne de score maximum de façon gloutonne et en exploitant les contraintes restantes pendant l'algorithme, voire a posteriori. Pour cela, un premier chemin de score maximum est cherché, puis toutes les ancres du chemin trouvé sont retirées du graphe de relation R. La procédure est répétée tant que les chaînes trouvées ont un score considéré comme suffisant. La suppression des ancres entre chaque étape permet de garantir que la contrainte 3 est satisfaite. Les autres contraintes sont ignorées ou font l'objet d'un filtrage a posteriori (en éliminant les chaînes qui les violent). Cette approche gloutonne ne garantie naturellement pas de construire un ensemble de chaînes dont le score total, sur toutes les chaînes, est maximum car les chaînes ne sont pas indépendantes les unes des autres et la maximisation du score d'une chaîne se fait au détriment du score des autres chaînes de l'ensemble, comme le montre un simple exemple :

Avec quatre chaînes c_i, c_j, c_k et c_l de score $S(c_i) = s$, $S(c_j) = -(s-2)$, $S(c_k) = s-1$ et $S(c_l) = 1$. Si c_i, c_j, c_k se suivent, alors il existe une chaîne c de score $S(c) = s+1$ construit à partir de ces trois chaînes. L'approche gloutonne qui maximise le score de la chaîne courante donnera : $\mathcal{C} = \{c, c_l\}$ avec $S(\mathcal{C}) = s+2 \approx s$. Il existe naturellement un ensemble de chaîne de meilleur score $\mathcal{C}' = \{c_i, c_k\}$ avec un score $S(\mathcal{C}') = 2 \times s - 1 \approx 2 \times s$. L'approche gloutonne a de plus tendance à construire des chaînes qui peuvent contenir des sous segments (ici c_j de très mauvaise qualité).

En pratique, tant que le graphe de relation est peu dense (avec un ratio d'arcs/sommets proche de 1), alors il y a sans doute peu de différences entre la maximisation gloutonne du score des chaînes et la maximisation du score de l'ensemble des chaînes (il y a peu de compétition pour les sommets/ancres).

Cependant les choses sont différentes lorsque le graphe est dense. Notre graphe de relations, qui permet de capturer des mouvements évolutifs variés, crée un graphe plus dense que dans les précédents outils [9, 3, 8, 14, 15], et une optimisation plus globale du score de l'ensemble des chaînes prend alors de l'importance.

2.8 Fonctions de score

Pour différencier les ancres et les chemins les plus intéressants *a priori*, un score est associé à chaque sommet et à chaque arc. En ce qui concerne les ancres, nous nous appuyons sur le fait que ces ancres sont généralement détectées par des logiciels dédiés fournissant une e-value (l'espérance du nombre de régions similaires de cette qualité qui devraient exister du seul fait du hasard). Seules les ancres de e-value très faible (inférieure à 10^{-4}) sont habituellement conservées. Une ancre a_i de e-value $a_i.s$ se voit attribuée un score

$$S(a_i) = \min\{-log_{10}(a_i.s), 250\}$$

Le seuil de 250 permet de borner les scores extrêmes générés par des probabilités considérées comme nulles sur un flottant. On favorise donc la création de chaînes contenant des ancres solides.

En ce qui concerne les régions séparant les ancres, le score choisi favorise les paires d'ancres séparées par une distance courte. Pour favoriser les ancres dont les positions relatives sont cohérentes avec une duplication simultanée (sans insertion ou délétion supplémentaire), la distance choisie favorise les diagonales. Ainsi, pour deux ancres a_i et a_j séparées par $\Delta^{\mathsf{A}} = (a_j^{\mathsf{A}}.deb - a_i^{\mathsf{A}}.fin)$ et $\Delta^{\mathsf{B}} = (a_j^{\mathsf{B}}.deb - a_i^{\mathsf{B}}.fin)$, le score utilisé est :

$$S(a_i \prec a_j) = -\Big(\Delta^{\mathsf{A}} + \Delta^{\mathsf{B}} + |\Delta^{\mathsf{A}} - \Delta^{\mathsf{B}}|\Big)/2 \tag{7}$$

La distance corrrespondante est égale à la distance euclidienne pour les horizontales ou les verticales sur un dotplot mais divise la distance euclidienne par $\sqrt{2}$ pour les diagonales.

3 Algorithme

Nous cherchons un ensemble de n chemins dans R, ne partageant aucun sommet, et de score global maximum.

Pour résoudre efficacement ce problème fortement combinatoire (le simple nombre de chemins dans le graphe de relations peut croître de manière exponentielle avec la taille du graphe), nous exploitons une relation forte entre flots dans un réseau où toutes les capacités sont de 1 d'une part et ensembles de chemins reliant la source du réseau au puits d'un réseau d'autre part. Nous montrons ainsi que la recherche d'un ensemble de chemins ne partageant aucun sommet et de score maximum dans le graphe de relation peut se ramener à la recherche d'un flot de coût minimum dans un réseau dérivé du graphe de relations où les coûts des arcs seront égaux à l'opposé des scores du graphe d'origine.

Nous rappelons les définitions essentielles (réseau, flot, valeur d'une flot, coût d'un flot...).

3.1 Réseau et flot de coût minimum

Le problème de flot de coût minimum est défini sur un *réseau de transport* :

Définition 3 (Réseau de transport) *Un réseau de transport est un graphe orienté* $T = (S, A, C, W, \mathbf{s}, \mathbf{s})$ *dans lequel* S *et* A *représentent l'ensemble des sommets et arcs du graphe. Pour chaque arc* $a = (i, j) \in A$*, on dispose de la capacité de transport* $C(a) \in \mathbb{N}$*, notée aussi* c_{ij} *et du coût de transport* $W(a)$ *d'une unité de flot, aussi notée* w_{ij}*.* S *contient 2 sommets identifiés* $\mathbf{s}$ *(resp.* $\mathbf{p}$*) de degré entrant (resp. sortant) égal à zéro.*

Pour $u \in S$, on note $\Gamma^+(u)$ (resp. $\Gamma^-(u)$) l'ensemble des arcs ayant pour origine (resp. extrémité) u.

Définition 4 (Flot, valeur, coût) *Dans un réseau de transport* T*, un flot est une application* $f : A \to \mathbb{N}$ *(nous noterons* f_{ij} *la valeur du flot sur l'arc* (i, j)*) vérifiant :*

1. *Capacité :* $\forall (i, j) \in A, \quad 0 \leq f_{ij} \leq c_{ij}$
2. *Conservation du flot :* $\forall i \in S, i \notin \{\mathbf{s}, \mathbf{p}\} \quad \sum_{j \in \Gamma^-(i)} f_{ji} = \sum_{j \in \Gamma^+(i)} f_{ij}$
3. *Flot puits/source :* $\sum_{j \in \Gamma^+(\mathbf{s})} f_{\mathbf{s}j} = \sum_{j \in \Gamma^-(\mathbf{p})} x_{j\mathbf{p}} = V(f)$

$V(f)$ *est appelée la valeur du flot* f*. On définit le coût d'un flot* f *comme* $C(f) = \sum_{(i,j) \in A} w_{ij}.f_{ij}$.

Soit $\mathbf{F}$ l'ensemble des flots de valeur n, le problème de recherche d'un flot de valeur n de coût minimum consiste à trouver un flot f

$$f = arg \min_{f \in \mathbf{F}} C(f) \tag{8}$$

3.2 Représentation du graphe de relation sous forme de réseau

Dans un premier temps, nous transformons le graphe de relation R introduit dans la section précédente en un réseau de transport T_R. Avant toute chose, tous les scores du graphe R sont transformés en un coût de valeur opposée afin de ramener le problème de maximisation de score à un problème de minimisation de coût.

Les sommets du graphe de relation étant porteurs d'une information de coût, nous transformons, de façon classique, tout sommet a_i de R en une paire de sommets a_i et a'_i reliés par un arc allant de a_i à a'_i. Le coût $w_{ii'}$ associé à cet arc est égal à $-S(a_i)$.

Enfin, les sommets $\mathbf{s}$ et $\mathbf{p}$ sont ajoutés. $\mathbf{s}$ est relié à tous les sommet a_i de R par un arc de coût $w_{\mathbf{s}i} = 0$. De même, tous les sommets $a_{i'}$ ajoutés à l'étape précédente sont reliés à $\mathbf{p}$ par un arc de coût $w_{i'\mathbf{p}} = 0$. Toutes les capacités sont fixées à 1.

Propriété 1 *Dans* T_R*, tout flot entier* f *de valeur* n *et de coût* $C(f)$ *correspond à un ensemble de chemins ne partageant aucun sommet dans* R *et de score* $-C(f)$*.*

Preuve : Soit un flot entier de valeur n dans T_R, dont tous les arcs ont une capacité de 1. Chaque sommets a_i de R étant représenté par un arc dans T_R, chaque sommet de T_R (en dehors de $\mathbf{s}$ et $\mathbf{p}$) a soit un degré entrant égal à 1, soit un degré sortant égal à 1. Il reçoit donc une unité de flot d'un seul de ses arcs entrants et la diffuse sur un seul des arcs sortants. Globalement, le flot de valeur n est donc formé d'un ensemble de n chemins $(\mathbf{s}, a_{i_1}, a_{i'_1}, \ldots, \mathbf{p})$ ne partageant aucun arc entre eux. Les sommets a_i de R étant représentés par un arc dans T_R, après suppression des sommets $a_{i'}$, ces n chemins définissent n chemins ne partageant aucun sommet entre eux dans R.

Réciproquement, pour tout chemin $(a_{i_1}, \ldots, a_{i_k})$ issu d'un ensemble donné de n chemins ne partageant pas de sommets dans R, on peut construire un chemin $(\mathbf{s}, a_{i_1}, a_{i'_1}, \ldots, a_{i_k}, a_{i'_k}, \mathbf{p})$ dans T_R. Les chemins obtenus ne partagent aucun arc et l'ensemble de ces arcs définit donc un flot de valeur n dans T_R.

Dans les deux cas, le score de l'ensemble des chemins, égal à la somme des scores des arcs et sommets traversés dans R est bien égal à l'opposé du coût du flot dans T_R (le flot étant d'une unité sur tout arc porteur de flot). □

On constate donc qu'il suffit de trouver un flot de coût minimum dans T_R pour trouver un ensemble de chemins de score optimal et ne partageant aucun sommet (ancre) dans R. L'algorithme de Busacker et Gowen [2, 5, 6, 7] permet de calculer un flot maximum de coût minimum par une procédure itérative. Dans notre cas, le flot maximum ne nous intéresse pas car il correspond à un flot de valeur $|A|$, traversant chaque ancre a_i indépendemment. Nous avons donc adapté l'algorithme de Busacker et Gowen. Dans notre cas, cet algorithme augmente la quantité de flot de 1 à chaque itération.

Propriété 2 *À chacune des itérations de l'algorithme, un flot de coût minimum est fourni par l'algorithme de Busacker et Gowen.*

Cette propriété, démontrée par Gowen et Busacker [2] et Giroudeau [11], nous permet d'obtenir, à l'itération n, un flot de valeur n et de coût minimum. En pratique, la graphe de relation R n'étant pas connexe, nous appliquons la transformation précédente à chaque composante connexe et obtenons m *réseaux de transport*, où m est le nombre de composantes connexes de R. A chaque itération, le réseau de transports contenant un chemin de coût minimum voit son flot augmenter. La compléxité d'une itération est de $O(sr)$ avec s le nombre de sommets et r le nombre d'arcs de la plus grande composante connexe et l'algorithme est en $O(nsr)$.

3.3 Les chaînes non valides

Contrairement à l'approche gloutonne, l'approche par flot de coût minimum présente l'avantage d'intégrer l'optimisation d'un score global, la prise en compte de la contrainte 1 de proximité des ancres (via le graphe R) mais aussi la condition 3 de non partage des ancres entre chaînes (via les capacités unitaires) et la condition 6 (dans le cas des duplications segmentales, en éclaircissant le graphe R).

Elle n'empêche cependant pas de produire des chaînes violant les conditions de non chevauchement 4 et 5 (dans le cas des duplications segmentales). Dans la pratique, certains outils, comme OSFinder [15], suppriment toutes les ancres intersectant le rectangle défini par la dernière chaîne créée dans un processus glouton (mais cette condition n'est cependant pas suffisante pour éviter totalement le problème : un arc peut encore chevaucher le rectangle). Dans DAGchainer [8], la violation de la contrainte 4 est négligée et les chaînes qui ne respectent pas la condition 5 sont déclarées en *tandem* et supprimées. DAGchainer viole aussi la contrainte 3 en exécutant deux passes de son algorithme, la première (resp. la deuxième) pour trouver les chaînes de polarités positives (resp. négatives).

Dans notre cas, les chaînes produites par l'algorithme et qui violent les conditions 4 et 5 sont classés en deux catégories. On distingue ainsi les chaînes dites “non valides temporaires” des autres. Ces chaînes sont non valides, mais contiennent au moins une sous-chaîne qui satisfait les conditions 4 et 5.

L'algorithme de Busaker et Gowen poursuit alors ses itérations (et la valeur du flot est incrémentée)

tant que l'une des deux conditions suivantes est satisfaite : (1) les chemins produits par l'algorithme ont un coût supérieur à un paramètre *minScore* et (2) il existe dans les chemins représentés dans le flot courant un chemin *non valide temporaire*. Le paramètre *minScore* définit le score minimum attendu d'un chemin.

A la différence des algorithmes gloutons employés traditionnellement dans le domaine, notre algorithme produit donc de façon garantie un ensemble de chemins vérifiant les équations 3 (une ancre appartient à un seul chemin) et 4 (non chevauchement sur les deux génomes). La première propriété est garantie par l'algorithme de flot, et la seconde par la deuxième partie du critère d'arrêt utilisé (existence de chemins "non valides temporaires").

La seule propriété qu'il reste à vérifier est donc la propriété 2, exigeant que les chemins formés définissent des régions suffisamment "solides" (formées d'un nombre suffisant d'ancres, sans arc trop coûteux et raisonnablement linéaires). Notre implémentation se contente de filtrer toutes les chaînes violant cette propriété.

3.4 Estimation des paramètres

Afin de rendre l'algorithme plus robuste et facile à paramétrer, on normalise d'abord les scores des sommets et les scores des arcs indépendemment. Le score moyen des arcs est fixé arbitrairement à $-1\ 000$ (pour rester entier) et celui des sommets à $1\ 000.\alpha$ où α est un paramètre de contraste entre ancres (sommets) et arcs les reliant. Par la suite α est fixé à 5.

L'ensemble de chemins retournés par l'algorithme dépend alors encore de quatre paramètres : $DistMax_{\mathsf{A}}, DistMax_{\mathsf{B}}, minAncre, minScore$. Les paramètres $DistMax_{\mathsf{A}}, DistMax_{\mathsf{B}}$ sont estimés à l'aide de l'équation 9 ci-dessous, appliquée aux données fournies en entrée. Soit D_g la somme des longueurs entre deux ancres consécutives sur le génome g :

$$DistMax_g = \frac{D_g}{\sqrt{2|A|}} \tag{9}$$

$D_g/\sqrt{|A|}$ constitue une approximation de la distance euclidienne moyenne entre deux ancres distribués aléatoirement sur la région bidimensionelle d'un dotplot [14]. Le facteur $\sqrt{2}$ tient compte de la distance particulière utilisée dans notre cas.

Enfin, le score minimum d'un chemin *minScore* est lié, via un paramètre β, au score moyen d'un plus petit chemin acceptable (formé de *minAncre* ancres de score moyen $1000.\alpha$ et de $minAncre - 1$ arcs de score moyen 1000 les reliant) :

$$minScore = \beta.\Big(\alpha.1000.minAncre + 1000.(minAncre - 1)\Big) \tag{10}$$

Dans la suite β est fixé à 4, le nombre minimum d'ancre dans une chaîne $minAncre = 3$ et le coefficient de corrélation minimum $minCorr = 0,8$.

4 Évaluation expérimentale

Nous avons évalué notre algorithme dans le cas de la recherche de duplications segmentales sur le génome d'*Arabidopsis thaliana* et de régions homologues entre le génome de *Glycine max* et de *Medicago truncatula*. Sur chacun des cas, nous présentons d'abord une comparaison entre la méthode gloutonne et la méthode par flot de coût minimum, toutes choses étant égales par ailleurs (fonctions

Arabidopsis vs Arabidopsis		Score	Coef. de corrélation			Longueur en kb		
Méthode	Nombre de chaînes		min	moy	max	min	moy	max
Gloutonne	100	$21,2 \cdot 10^6$	0,15	0,93	1	231	2341	8968
	200	$27,7 \cdot 10^6$	0,02	0,82	1	61	1545	8968
	300	$32,6 \cdot 10^6$	0,02	0,84	1	41	1261	8968
Flot	100	$21,6 \cdot 10^6$	0,15	0,94	1	669	2540	8150
	200	$28,4 \cdot 10^6$	0,12	0,91	1	85	1741	8150
	300	$33,5 \cdot 10^6$	0,01	0,87	1	60	1435	8968

TAB. 1 – Différences au niveau du score, du cofficient de corrélation et de la longueur des chaînes entre la méthode *gloutonne* et la méthode de *flot*, dans le cas de la recherche de duplications segmentales sur le génome *Arabidopsis thaliana*.

de scores, graphe de relation...). Dans un second temps, sur les même génomes, nous comparons l'outil ReD qui implémente la méthode par flot de coût minimum avec les logiciels DAGchainer [8] et OSfinder [15]. DAGchainer est le logiciel le plus cité utilisé dans les analyses génomiques. Nous y avons ajouté OSfinder en tant que logiciel très récent.

Pour alimenter ces outils, il faut détecter les ancres. Pour cela nous avons utilisé le logiciel BLASTP [1] en comparant les protéines du génomes A aux protéines de génomes B en ne conservant que les résultats ayant une *e-value* de 10^{-5} au plus. Pour réduire les lignes verticales et horizontales qui bruitent le dotplot, un filtre est ajouté pour supprimer toutes les paires de gènes qui possèdent au moins un gène présent dans plus de 50 (par défaut) paires de gènes.

4.1 Comparaison avec la méthode gloutonne

L'approche par flot et gloutonne ont été comparées sur le problème de la recherche d'un nombre de châines fixées, en ignorant les contraintes de validité (2, 4 et 5). Dans tous les cas, le graphe de relation R, les fonctions de score sont identiques. Les performances sont comparées à l'aide de trois critères : le *score* de l'ensemble de chaînes produit ; le *coefficient de corrélation* évaluant la linéarité des châines produites et la *longueur* totale des chaînes (somme des longueurs des intervalles couvrant les deux séquences génomiques). Un score élevé, une forte linéarité et une large couverture des génomes est préférable sachant que ces critères sont souvent antagonistes.

Les résultats, présentés dans les tableaux 1 et 2 montrent que l'approche gloutonne fournit des scores inférieurs et que la différence entre les deux approches augmente avec le nombre de chaînes construites. L'approche par flot de coût minimum permet d'autre part de construire des chaînes plus longues (ajout d'ancres aux extrémités des chaînes) avec une meilleure linéarité : les choix locaux réalisés par l'approche gloutonne pénalisent la qualité globale des chaînes.

4.2 Comparaison avec des logiciels existants

La comparaison avec des logiciels existants n'est pas évidente. Tout d'abord parce qu'il n'existe pas de jeux de données où la vérité est connue. Mais aussi car chaque logiciel intègre ses propres méthodes

Glycine vs Medicago		Score	Coef. de corrélation			Longueur en kb		
Méthode	Nombre de chaînes		min	moy	max	min	moy	max
Gloutonne	200	$21,9 \cdot 10^6$	0,07	0,93	1	261	1951	16900
	400	$30,0 \cdot 10^6$	0,00	0,92	1	233	1357	16900
	600	$36,0 \cdot 10^6$	0,00	0,90	1	159	1092	16900
Flot	200	$21,9 \cdot 10^6$	0,08	0,95	1	255	2051	16900
	400	$30,2 \cdot 10^6$	0,10	0,93	1	242	1464	16900
	600	$36,3 \cdot 10^6$	0,00	0,92	1	159	1197	16900

TAB. 2 – Différences au niveau du score, du cofficient de corrélation et de la longueur des chaînes entre la méthode *gloutonne* et la méthode de *flot*, dans le cas de la recherche de régions homologues entre le génome de *Glycine max* et et le génome de *Medicago truncatula*.

de filtrage pour les données en entrée, des fonctions de score différentes voire même un graphe de relation différent. Pour évaluer les logiciels, en dehors des mesures de longueur et de corrélation déjà réalisés, nous nous sommes appuyés également sur la caractère cohérent des ensembles de chaînes produits. En effet, la succession d'évènements de spéciation et de duplication ne peuvent pas créer deux relations d'homologies qui se chevauchent sur les deux séquences génomique (condition 4). Les chaînes violant cette condition sont comptabilisées dans la colonne *Chevauche*. Dans le cas de la comparaison d'un génome contre lui-même, une relation d'homologie entre deux régions qui s'intersectent viole également la condition 5. Dans ce cas, les chaînes sont proches de la diagonale du dotplot et sont donc comptabilisées dans la colonne *Diag*.

Arabidopsis vs Arabidopsis	Coef. de corrélation			Longueur en kb			nombres de chaînes		
Logiciel	min	moy	max	min	moy	max	Total	Diag	Chevauche
DAGchainer	0,55	0,99	1	27	1176	8663	306	65	63
OSfinder	0,23	0,93	1	7	4784	60430	286	5	126
ReD	0,80	0,95	1	24	734	3618	281	0	0

TAB. 3 – Ce tableau montre le nombre d'ancres utilisés par les trois logiciels, ainsi que le nombres de chaînes crées et le nombres de chaînes incohérents, dans le cas de la recherche de duplications segmentales sur le génome *Arabidopsis thaliana*.

Les tableaux 3 et 4 nous montrent que seul ReD crée un ensemble de chaînes cohérent alors que la proportion de chaînes incohérentes crées par DAGchainer et OSfinder est comprise entre 10% et 50%. Il est naturellement possible de filtrer ces chaînes, mais le nombre de chaînes diminue d'autant. Une alternative consiste à trouver un sous-ensemble de chaînes cohérent mais ce n'est pas un problème simple, les chaînes incompatibles s'excluent mutuellement. ReD règle se problème en amont, dans son formalisme, et dans l'algorithme de flot associé.

Les sorties de ReD possèdent un coefficient de corrélation un peu plus faible que celle de DAGchainer. Cette différence s'explique par le fait que ReD a la possibilité de créer des chaînes qui ne respectent pas la contrainte d'ordre (section 2.4) ce qui peut pénaliser la linéarité (comme nous le montre la figure 3), cependant la différence reste faible.

Glycine vs Medicago	Coef. de corrélation			Longueur			nombres de chaînes	
Logiciel	min	moy	max	min	moy	max	Total	Chevauche
DAGchainer	0,77	0,99	1	7	976	16900	651	71
OSfinder	0,27	0,97	1	3	2766	29830	7864	2143
ReD	0,83	0,98	1	135	1179	6745	418	0

TAB. 4 – Ce tableau montre le nombre d'ancres utilisés par les trois logiciels, ainsi que le nombre de chaînes crées et le nombre de chaînes incohérentes, dans le cas de la recherche de régions homologues entre le génome de *Glycine max* et et le génome de *Medicago truncatula*.

Les résultats d'OSfinder sont sensiblement moins bons, avec des valeurs extrêmes en terme de longueur des chaînes et de leur nombre. Cette différence est sans doute due au fait qu'OSFinder a été développé pour détecter les régions homologues seulement et non les duplications segmentales et testé sur des génomes qui possèdent moins de duplications que les génomes de plantes utilisés.

5 Conclusion

Via l'utilisation jointe d'une modélisation sous la forme d'un graphe orienté pondéré et d'un algorithme de recherche de flot de coût minimum dans un graphe de transport idoine, l'outil ReD permet donc d'identifier des régions homologues entre génomes avec des garanties en termes de score et de respect des constraintes qui dépasssent celles des outils existants.

Le passage d'une méthode gloutonne à une méthode de recherche de flot, plus complexe, reste tout à fait comfortable en termes de temps de calcul : sur une machine récente et avec un graphe de relations comportant 165977 ancres et 487932 arcs, l'implémentation actuelle de ReD, en C++, prend de l'ordre de 15 minutes pour s'exécuter. À la différence des outils existants, ReD permet d'activer ou non les contraintes optionelles d'ordre et d'orientation présentées dans la section 2.4 et reste le seul outil prenant en compte les contraintes spécifiques de non chevauchement qui apparaissent dans la détection de duplications segmentales. Les sorties de ReD ont été intégrées à l'outil de visualisation Narcisse [4] qui permet de visualiser et d'analyser les résultats de l'analyse des duplications segmentales dans plusieurs génomes de plantes sous forme graphique.

Pour autant, ReD n'est encore qu'une étape sur la voie qui mène à l'outil idéal de recherche de régions homologues. La prise en compte des contraintes de non chevauchement dans le critère d'arrêt n'est pas toujours idéale et peut parfois mener à une fragmentation excessive des chaînes dans les génomes fortement dupliqués. Enfin et surtout, il reste à évaluer les performances de ReD dans un contexte plus difficile et encore peu exploré : l'exploitation d'ancres détectées directement au niveau de l'ADN qui est largement plus bruitée qu'au niveau des protéines (utilisé traditionnellement pour détecter les ancres). ReD pourra alors être utilisé avant le processus d'annotation des génomes (détection des régions correspondant aux protéines) et ses sorties pourraient alors alimenter ce processus de prédiction.

Références

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. *Basic local alignment search tool.*, volume 215. National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894., October 1990.

[2] Gowen Paul J Busacker Robert G. A procedure for determining a family of minimum-cost network flow patterns. November 1960.

[3] Steven B Cannon, Alexander Kozik, Brian Chan, Richard Michelmore, and Nevin D Young. Diaghunter and genopix2d : programs for genomic comparisons, large-scale homology discovery and visualization. *Genome Biology*, pages 853–860, September 2003.

[4] E. Courcelle, Y. Beausse, S. Letort, O. Stahl, R. Fremez, C. Ngom-Bru, J. Gouzy, and T. Faraut. Narcisse : a mirror view of conserved syntenies. *Nucleic Acids Res*, 36(Database issue) :D485–90, 2008.

[5] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2) :248–264, 1972.

[6] Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3) :247–255, 1985.

[7] Andrew V. Goldberg and Robert E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. ACM*, 36(4) :873–886, 1989.

[8] B. J. Haas, A. L. Delcher, J. R. Wortman, and S. L. Salzberg. Dagchainer : a tool for mining segmental genome duplications and synteny. *Bioinformatics*, 20(18) :3643–3646, December 2004.

[9] Sugata Chakravarty Peter P. Calabrese and Todd J. Vision. Fast identification and statitistical evaluation of segmental homologies in comparative maps. *Bioinformatics*, 19 :i74–i80, January 2003.

[10] Pavel Pevzner and Glenn Tesler. Genome rearrangements in mammalian evolution : Lessons from human and mouse genomes. *Genome Research*, 13(1) :37–45, January 2003.

[11] R. Giroudeau. *Cours Algorithmique/complexité/Calculabilité*. 2009. Cours de Master, Univ. de Montpellier II. Disponible sur http ://www.lirmm.fr/~rgirou/enseignement/pageenseignement.html.

[12] C. Simillion, K. Vandepoele, Y. Saeys, and Y. Van de Peer. Building genomic profiles for uncovering segmental homology in the twilight zone. *Genome research*, 14(6) :1095–1106, June 2004.

[13] Amit U. Sinha and Jaroslaw Meller. Cinteny : flexible analysis and visualization of synteny and genome rearrangements in multiple organisms. *BMC Bioinformatics*, 8 :82+, March 2007.

[14] Carol Soderlund, William Nelson, and Austin Shoemaker et al. Symap : A system for discovering and viewing sytenic regions of fpc maps. *Genome research*, 16 :1159–1168, December 2006.

[15] Kris Popendorf Tsuyoshi Hachiya, Yasunori Osana and Yasubumi Sakakibara. Accurate identification of orthologous segments among multiple genomes. *Bioinformatics*, 25, February 2009.

[16] K. Vandepoele, Y. Saeys, C. Simillion, J. Raes, and Y. Van De Peer. The automatic detection of homologous regions (adhore) and its application to microcolinearity between arabidopsis and rice. *Genome Res*, 12(11) :1792–1801, November 2002.