Sous la co-tutelle de :
CNRS
ÉCOLE DES PONTS PARISTECH
UNIVERSITÉ GUSTAVE EIFFEL

Paris-Est Sup
Laboratoire d'Informatique Gaspard-Monge (UMR 8049)

Proximité, similarité et hérédité : de la bioinformatique aux humanités numériques

Dossier de travaux

Présenté par Philippe Gambette

pour une candidature à l'habilitation à diriger des recherches en INFORMATIQUE

22 aout 2024

Les pages suivantes réunissent dix de mes articles mentionnés dans la synthèse de mes travaux de recherche intitulée *Proximilarity, similarity and heredity : from bioinformatics to digital humanities* (*Proximité, similarité et hérédité : de la bioinformatique aux humanités numériques*), dans leur version mise à disposition dans les archives ouvertes HAL ou ArXiv, que je peux donc diffuser dans ce document [6, 9, 8, 7, 3, 5, 4, 1, 10, 2].

Mes publications sont toutes fournies en libre accès sur HAL à l'adresse https://cv.hal.science/philippe-gambette, où les liens vers les publications sur le site des maisons d'édition sont fournis.

Je mets parfois à disposition quelques compléments (matériel supplémentaire : données, démo web ou code source) sur la page https://igm.univ-mlv.fr/ gambette/RePublications.php.

Bibliographie

[1] Rachel Bawden et al. “Automatic Normalisation of Early Modern French”. In : *LREC 2022 - 13th Language Resources and Evaluation Conference*. European Language Resources Association. Marseille, France, juin 2022. doi : 10.5281/zenodo.5865428. url : https://hal.inria.fr/hal-03540226.

[2] Pierre Bourhis, Aaron Boussidan et Philippe Gambette. “On Distances between Words with Parameters”. In : *CPM 2023*. Sous la dir. de Laurent Bulteau et Zsuzsanna Lipták. T. 259. Proceedings of the 34th Annual Symposium on Combinatorial Pattern Matching. Champs-sur-Marne, Marne-la-Vallée, France : Schloss Dagstuhl, juin 2023, 6:1-6:23. doi : 10.4230/LIPIcs.CPM.2023.6. url : https://hal.science/hal-04080842.

[3] Mathilde Bouvel, Philippe Gambette et Marefatollah Mansouri. “Counting phylogenetic networks of level 1 and 2”. In : *Journal of Mathematical Biology* 81 (oct. 2020), p. 1357-1395. doi : 10.1007/s00285-020-01543-5. url : https://hal-upec-upem.archives-ouvertes.fr/hal-02955527.

[4] Laurent Bulteau, Philippe Gambette et Olga Seminck. “Reordering a tree according to an order on its leaves”. In : *CPM 2022*. T. 223. LIPIcs. Prague, Czech Republic : Schloss Dagstuhl - Leibniz-Zentrum für Informatik, juin 2022, 24:1-24:15. doi : 10.4230/LIPIcs.CPM.2022.24. url : https://hal-upec-upem.archives-ouvertes.fr/hal-03413413.

[5] Chuanming Dong, Philippe Gambette et Catherine Domingùès. “Extracting Event-related Information from a Corpus Regarding Soil Industrial Pollution”. In : *KDIR 2021*. T. 1. 13th International Conference on Knowledge Discovery and Information Retrieval. Setúbal, Portugal : SciTePress, oct. 2021, p. 217-224. doi : 10.5220/0010656700003064. url : https://hal.archives-ouvertes.fr/hal-03366097.

[6] Philippe Gambette, Núria Gala et Alexis Nasr. “Longueur de branches et arbres de mots”. In : *Corpus* 11.- (2012), p. 129-146. url : https://hal-upec-upem.archives-ouvertes.fr/hal-00822993.

[7] Philippe Gambette et al. “Anatomie, animaux, vocabulaire de la vivisection”. In : *Animalhumanité - Expérimentation et fiction : l’animalité au cœur du vivant*. Sous la dir. de Gisèle Séginger. Savoirs en Texte. LISAA, 2018, p. 223-231. url : https://hal.archives-ouvertes.fr/hal-01609198.

[8] Philippe Gambette et al. “Do branch lengths help to locate a tree in a phylogenetic network?” In : *Bulletin of Mathematical Biology* 78.9 (2016), p. 1773-1795. doi : 10.1007/s11538-016-0199-4. url : https://hal-upec-upem.archives-ouvertes.fr/hal-01372824.

[9] Philippe Gambette et al. “Locating a Tree in a Phylogenetic Network in Quadratic Time”. In : *RECOMB 2015*. T. 9029. LNCS. Varsovie, Poland : Springer, avr. 2015, p. 96-107. doi : 10.1007/978-3-319-16706-0_12. url : https://hal-upec-upem.archives-ouvertes.fr/hal-01116231.

[10] Olga Seminck et al. “The Evolution of the Idiolect over the Lifetime: A Quantitative and Qualitative Study of French 19th Century Literature”. In : *Journal of Cultural Analytics* 7.3 (2022). doi : 10.22148/001c.37588. url : https://hal.science/hal-03767854.

Longueur de branches et arbres de mots

Philippe Gambette, Núria Gala, Alexis Nasr

Longueur de branches et arbres de mots

Philippe Gambette[1], Nuria Gala[2], Alexis Nasr[2]
[1]Université Paris-Est – LIGM, [2]Université Aix-Marseille – LIF

Résumé : Les arbres de mots constituent un des outils de la statistique textuelle pour visualiser les relations sémantiques entre mots d'un texte. Les méthodes de construction de ces arbres à partir d'une distance de co-occurrence dans le texte produisent des arbres dont les longueurs d'arêtes se prêtent mal à l'analyse. Pour faciliter l'interprétation visuelle de l'arbre, l'idéal serait que des longues arêtes séparent des classes sémantiques de mots. Ainsi, découper les arêtes les plus longues de l'arbre devrait conduire à une partition de l'ensemble des mots qui fournit des classes pertinentes. À l'aide de deux corpus dont un sous-ensemble de mots a été partitionné en un ensemble de classes sémantiques, nous évaluons plusieurs formules permettant de recalculer les longueurs d'arêtes de l'arbre construit à partir des distances de co-occurrence, afin de rendre l'interprétation de l'arbre plus facile et plus fiable.

Mots-clés : classification hiérarchique, visualisation, arbre, nuage arboré, co-occurrence, partition

Branch Lengths and Word Trees

Philippe Gambette[1], Nuria Gala[2], Alexis Nasr[2]
[1]Université Paris-Est – LIGM, [2]Université Aix-Marseille – LIF

Summary: Word trees are one of the available tools in textual analysis to visualize semantic relationships between the words of a text. Tree construction methods from the co-occurrence distances between words in a text produce trees whose edge lengths are difficult to analyze. In order to make the visual interpretation of the tree easier, long edges should separate semantic classes of words. Therefore, cutting the longest edges in the tree should lead to a partition of the word set with relevant classes. Using two corpuses where a subset of words was partitioned into semantic classes, we evaluate several formulas computing new edge lengths for a tree built from co-occurrence distances, aiming at making the interpretation of the tree easier and more reliable.

Keywords: hierarchical clustering, visualization, tree, tree cloud, co-occurrence, partition

Longueur de branches et arbres de mots

Philippe Gambette[1], Nuria Gala[2], Alexis Nasr[2]
[1]Université Paris-Est – LIGM, [2]Université Aix-Marseille – LIF

1. Introduction

1.1 Les arbres de mots comme classifications

Les arbres de mots se sont ajoutés aux projections et aux réseaux de co-occurrence parmi les outils développés pour l'analyse textométrique des textes (Luong, 1989 ; Mayaffre, 2008). Ils permettent en effet de représenter de manière esthétique un nombre limité de classes de mots emboîtées (en nombre linéaire par rapport au nombre de mots), tout en laissant la possibilité de faire varier les tailles de caractères des mots, par exemple dans les nuages arborés (Gambette & Véronis, 2009), dont une illustration est donnée en Figure 1.

L'arbre est construit à partir d'une matrice de distances entre les mots, en utilisant un algorithme de classification hiérarchique. La *distance de co-occurrence* entre deux mots *a* et *b* dans cette matrice de distances est proche de 0 si les mots apparaissent souvent à proximité dans le texte, et grande s'ils apparaissent rarement ensemble. Interprétée comme une distance sémantique en suivant le principe selon lequel le sens du mot provient de ses voisins (Firth, 1957), elle conduit à l'analyse suivante de l'arbre construit pour refléter au mieux cette distance : un sous-arbre regroupe des mots dont les distances sont petites comparées aux distances avec les mots du reste de l'arbre, donc ils apparaissent plus fréquemment ensemble dans le texte qu'avec des mots du reste de l'arbre. On en déduit qu'ils constituent une classe sémantique, et donc bien

souvent représentent une thématique du texte dont ils ont été extraits.

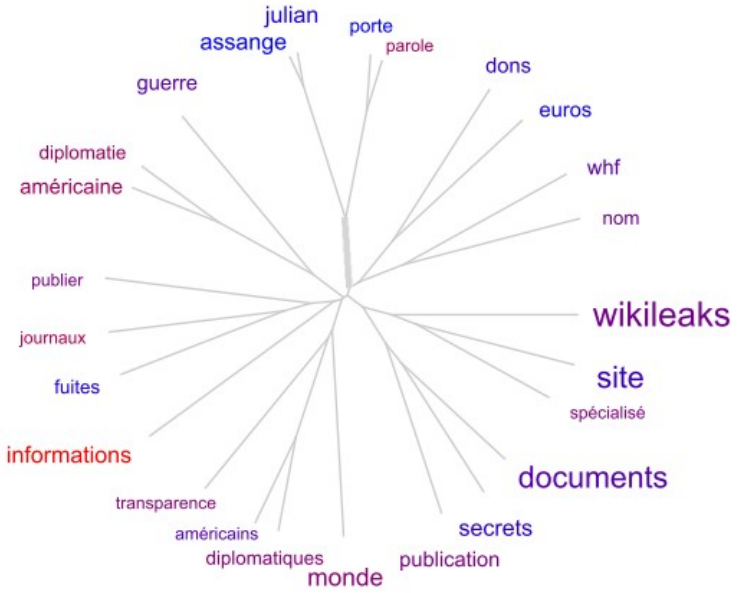


Figure 1 : Nuage arboré des 25 mots les plus fréquents (hors mots vides) du corpus Wikileaks (voir Section 4), construit par les logiciels TreeCloud (Gambette & Véronis, 2009) et SplitsTree (Huson & Bryant, 2006). L'arête en gras sépare la classe [julian, assange, porte, parole] du reste des mots.

On remarque par exemple, dans une lecture rapide du nuage arboré de la Figure 1, que le mot au coeur du texte utilisé pour construire cette visualisation est “wikileaks”, qui se trouve dans un sous-arbre aux côtés de “site” et “spécialisé”. Ceci nous permet d'esquisser une définition de la thématique principale du texte, que l'on complète en remarquant que le sous-arbre correspondant à la classe {wikileaks, site, spécialisé} est inclus dans le sous-arbre correspondant à la classe {wikileaks, site, spécialisé, publication, documents, secrets}. L'arbre rapproche également des composants du mot composé “porte-parole”, ou

le prénom “julian” du nom “assange”. Le fait que ces quatre mots sont les feuilles d'un même sous-arbre nous invite à déduire que dans le texte, Julian Assange est présenté comme le porte-parole de Wikileaks.

1.2 Interprétation des longueurs de branches

Cette interprétation d'un arbre de mots comme un simple ensemble de classes de mots emboîtées prend en compte uniquement la topologie de l'arbre, et ne fait pas intervenir les longueurs de branches. Pourtant, des longueurs de branches sont naturellement calculées par toute méthode de classification hiérarchique à partir d'une matrice de distance. La propriété attendue de ces longueurs est que les distances dans l'arbre obtenu[1] soient aussi proches que possible des distances fournies en entrée dans la matrice. Ainsi, des méthodes qui s'attachent à fournir des longueurs d'arêtes pertinentes peuvent avoir pour objectif une optimisation par les moindres carrés entre la matrice de distance fournie en entrée et les distances estimées dans l'arbre en sortie.

Ce type de méthodes, pour lesquelles les distances entre feuilles dans l'arbre calculé ont un grand intérêt, ont été particulièrement utilisées pour l'étude de l'évolution des espèces ou *phylogénie* (Felsenstein, 2004). En effet, la longueur du chemin entre deux feuilles de l'arbre a une interprétation directe : c'est la distance évolutive entre les deux espèces représentées par ces feuilles, qui se reflète généralement dans la distance entre leurs ADN.

Cette interprétation des distances entre feuilles de l'arbre est beaucoup moins pertinente pour un arbre de mots, pour trois raisons : de *modélisation*, de *fiabilité* et de *lisibilité*.

---

[1] rappelons que la distance entre deux feuilles d'un arbre est égale à la somme des longueurs des arêtes dans le chemin allant, dans l'arbre, d'une feuille à l'autre..

Tout d'abord, notons que dans un arbre phylogénétique, les sommets internes représentent des espèces ancestrales, et que les longueurs d'arêtes modélisent des distances d'évolution entre les espèces, ancestrales ou actuelles, correspondantes. En revanche, dans un arbre de mots, il est difficile d'interpréter les nœuds internes, et d'en déduire une façon naturelle d'interpréter la longueur d'une arête située entre deux nœuds internes de l'arbre.

L'absence d'un modèle d'évolution arborée implique aussi un problème de fiabilité : contrairement aux distances phylogénétiques qui ont généralement une structure proche d'une métrique d'arbre, les distances de co-occurrence entre mots peuvent être très éloignées de toute représentation arborée. Ainsi, l'approximation, fournie par l'arbre obtenu en sortie, des distances entre feuilles données dans la matrice en entrée, peut être très mauvaise, quand bien même on aurait calculé l'arbre optimal au sens de l'optimisation des moindres carrés.

Enfin, même si l'on choisit de ne pas interpréter les longueurs d'arêtes internes de l'arbre, et de se focaliser uniquement sur les distances entre feuilles, en espérant une certaine fiabilité, l'estimation visuelle de ces dernières pose un problème de lisibilité. Dans la Figure 1 par exemple, s'il est clair que dans l'arbre, “diplomatie” est plus proche d'“américaine” que de “publier”, il est en revanche difficile de comparer la distance entre “diplomatie” et “wikileaks” avec celle entre “dons” et “monde”. Un problème supplémentaire de lisibilité apparaît avec les données textuelles : la longueur excessive des branches menant aux feuilles (appelées *arêtes externes*) réduit la lisibilité de l'intérieur de l'arbre. Il s'agit là d'un défaut des méthodes de construction d'arbres à partir de distances, en particulier la méthode Neighbor-Joining de Saitou & Nei (1987) utilisée dans cet article : en raison de la structure très particulière des formules de co-occurrence de mots (Evert, 2005), on peut constater que la longueur des arêtes internes d'un

nuage arboré est souvent très petite par rapport à celle des arêtes menant aux feuilles.

Ces trois constats montrent les limites d'une utilisation des arbres de mots dont les longueurs d'arêtes sont calculées directement par l'algorithme de classification hiérarchique. Dès lors, il convient de proposer un modèle d'interprétation de l'arbre, et de calcul de ses longueurs d'arêtes, qui soit pertinent pour la textométrie.

Nous avons vu que l'interprétation la plus immédiate de l'arbre consiste à considérer chaque sous-arbre comme une classe de mots. Il convient donc de faciliter et de renforcer cette interprétation, en choisissant une méthode de calcul des longueurs d'arêtes compatible avec cet objectif. Nous proposons de recalculer les longueurs de branches après la construction de l'arbre, de telle manière qu'elles assurent sa lisibilité, tout en facilitant la lecture de l'arbre comme une partition en classes de l'ensemble des mots. Pour cela, nous proposons d'utiliser des formules proposées par Guénoche & Garreta (2002) pour évaluer la qualité des arêtes d'un arbre. Ces formules indiquent si les deux ensembles de mots séparés par une arête sont effectivement bien séparés d'après la matrice de distance. Ainsi, on attribuera à chaque arête une longueur proportionnelle à son score de qualité, et les arêtes les plus longues seront les plus discriminantes.

1.3 Évaluation par la construction automatique d'une partition

Afin d'évaluer la pertinence de ces formules, nous proposons un algorithme qui construit une partition d'un ensemble de mots, à partir d'un arbre de mots, en découpant successivement, dans l'ordre décroissant des longueurs, ses arêtes internes, jusqu'à un certain critère d'arrêt, comme illustré en Figure 2. Cet algorithme correspond donc à la démarche effectuée visuellement par l'utilisateur de l'analyse arborée, qui interprète

les arêtes les plus longues de l'arbre comme des séparations entre des classes de mots regroupés en raison de leur proximité sémantique.

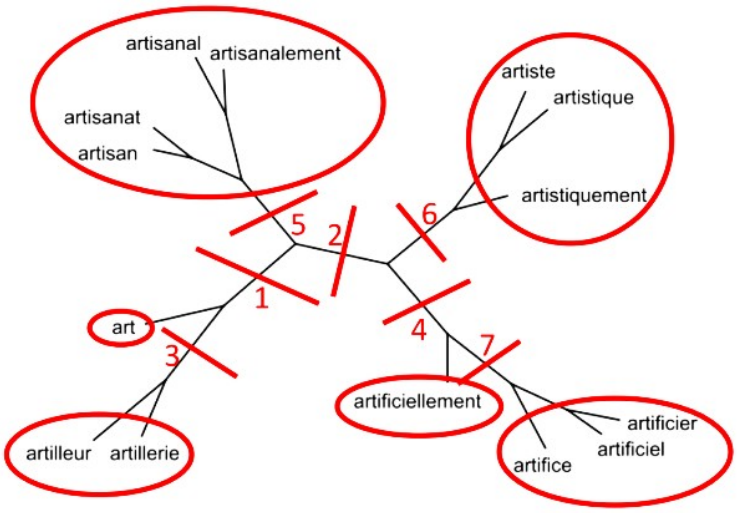


Figure 2 : Arbre de la famille de « art » dans la base de données Polymots (Gala & Rey, 2008). Les sept arêtes les plus longues sont numérotées dans l'ordre des longueurs décroissantes, et les six classes de mots obtenues après découpage de ces sept arêtes sont montrées par des ellipses rouges.

Ainsi, l'algorithme permet d'obtenir une partition de l'ensemble des mots, qui dépend directement de la longueur choisie pour les arêtes de l'arbre. C'est ce principe que nous allons utiliser pour évaluer les diverses formules permettant de calculer les longueurs d'arêtes de l'arbre. Nous allons en effet comparer des partitions de référence avec des partitions obtenues automatiquement par cette méthode.

Pour cela, nous utiliserons deux corpus d'évaluation. Le premier est constitué par 20 partitions de familles de mots de la base de données Polymots (Gala & Rey, 2008). Les

informations de distances entre les mots d'une même famille proviennent à la fois d'informations de co-occurrence dans le TLFi (Dendiel & Pierrel, 2003), et du nombre d'affixes communs (Gala et al., 2011). La partition de référence pour chacune de ces 20 familles a été construite manuellement, en effectuant des choix arbitraires de classe pour les mots polysémiques qui pourraient appartenir à plusieurs classes. Un exemple de famille et de sa partition en classes est donné au début de la Section 3.

Le second corpus est constitué par 10 textes écrits autour de 25 mots relatifs à l'organisation Wikileaks, ces mots étant organisés en une partition de référence. Chacun des 10 textes est donc censé faire apparaître les mots d'une même classe à proximité les uns des autres, et la distance de co-occurrence entre les mots est calculée par TreeCloud.

Ces deux corpus sont utilisés dans des protocoles d'évaluation qui permettent de faire émerger deux méthodes de calcul appropriées pour le calcul des longueurs d'arête de l'arbre, parmi les 5 testées : *triples* et *lengthRatio*.

2. Les formules de longueurs d'arêtes

Notre objectif est d'attribuer des longueurs d'arêtes cohérentes avec les informations de distance sémantique entre mots, c'est-à-dire une grande longueur aux arêtes qui séparent effectivement deux groupes de mots sémantiquement éloignés l'un de l'autre, et une petite longueur aux arêtes qui séparent des mots proches sémantiquement. Pour cela, plusieurs formules sont possibles à partir d'une matrice de distances entre mots de l'arbre (Guénoche & Garreta, 2002). Nous notons $d(a,b)$ la distance sémantique entre deux mots a et b d'après cette matrice de distances. Pour être précis, il ne s'agit pas exactement d'une distance au sens mathématique du terme, puisqu'elle ne respecte pas, généralement, l'inégalité triangulaire, mais d'une *dissimilarité*. Nous comparerons cinq formules possibles,

notées *computedLength*, *triples*, *quartets*, *lengthRatio*, et *agreementPairs*.

La première, *computedLength*, consiste à considérer simplement la longueur de l'arête calculée par l'algorithme de construction de l'arbre à partir de la matrice de distance, c'est-à-dire l'algorithme Neighbor-Joining (Saitou & Nei, 1987).

La deuxième, *triples*, désigne le taux de bons triplets séparés par l'arête. Elle consiste à calculer, pour tout ensemble de trois mots $\{a,b,c\}$, où a et b sont situés d'un côté de l'arête et c de l'autre côté, la proportion de deux qui vérifient :

$$d(a,b) \leq \min(d(a,c),d(b,c)).$$

Si l'arête est cohérente avec les données de distance sémantique, le score *triples*, compris entre 0 et 1, doit être proche de 1. Inversement, si les mots de part et d'autre de l'arête sont situés à distance inférieure aux mots d'un même côté de l'arête, ce score *triples* sera proche de 0.

De manière similaire, la troisième formule, *quartets*, désigne le taux de bons quadruplets séparés par l'arête. Pour le calculer, il faut évaluer, pour tout ensemble de quatre mots $\{a,b,x,y\}$, où a et b sont situés d'un côté de l'arête, et x et y de l'autre côté, la proportion de ceux qui vérifient :

$$d(a,b)+d(x,y) \leq \min(d(a,x)+d(b,y), d(a,y)+d(b,x)).$$

De nouveau, un bon score sera proche de 1 et un mauvais sera proche de 0.

Pour calculer le taux d'accord des paires, noté *agreementPairs*, on commence par classer, dans l'ordre croissant, les distances entre paires de mots distincts. Parmi ces $n(n-1)/2$ distances (pour n mots), on s'attend à ce que celles entre deux mots d'un même côté de l'arête soient inférieures à celles entre deux mots séparés par l'arête. En appelant donc m le nombre de paires de mots d'un même côté de l'arête, et d_m la m-ième plus petite distance entre paires de mots distincts, la formule *agreementPairs* calcule la somme du nombre de paires de mots distincts d'un même côté de l'arête dont la distance est

inférieure ou égale à *d_m* d'une part (conformément à ce qui est attendu), avec d'autre part le nombre de paires de mots distincts séparés par l'arête dont la distance est supérieure ou égale à *d_m* (conformément à ce qui est attendu), somme divisée par le nombre total de distances, soit $n(n\text{-}1)/2$. Comme il s'agit d'un taux de paires de mots, les scores des arêtes cohérentes avec les données de distance seront de nouveau proches de 1, et les mauvais proches de 0.

Enfin, le ratio des longueurs moyennes, noté *lengthRatio*, est la distance moyenne entre mots séparés par l'arête, divisée par la distance moyenne entre mots d'un même côté de l'arête. Si l'arête est bien cohérente avec les données de distance sémantique, on attend que ce score soit strictement supérieur à 1, sinon, il sera inférieur à 1.

3. Évaluation sur des familles morphologiques

3.1 Protocole d'évaluation

Pour chacune des formules de longueur d'arête décrites ci-dessus, nous avons testé leur performance avec la base de données Polymots. Cette base comprend 20 000 mots regroupés en 2 000 familles, chacune centrée autour d'un mot racine. Parmi ces familles, 20 ont été partitionnées manuellement en classes sémantiques. Par exemple, voici la partition pour un extrait de la base correspondant à la famille du radical « art » : [artificier, artifice, artificiel, artificiellement] [artillerie, artilleur] [artisan, artisanal, artisanalement, artisanat] [artiste, artistique, artistiquement, art].

Pour chacune des formules de longueur d'arêtes, nous effectuons, pour chaque famille de mots, une comparaison, à l'aide de l'indice de Rand, ou de l'indice de Rand corrigé, entre les partitions construites manuellement et celles construites automatiquement à partir d'une distance sémantique prenant en compte les co-occurrences des mots dans le TLFi ainsi que leur

nombre d'affixes communs (Gala et al, 2011). Les partitions construites automatiquement le sont en utilisant un algorithme de découpage des arêtes par longueur décroissante, illustré en Figure 2. Au *k*-ième découpage d'arête, on considère que chaque composante connexe obtenue, dans l'arbre ainsi découpé, fournit une classe de la partition. Nous obtenons ainsi une partition d'au plus *k*+1 classes, dont nous pouvons calculer un score de similarité avec la partition manuelle, comme montré en Figure 3.

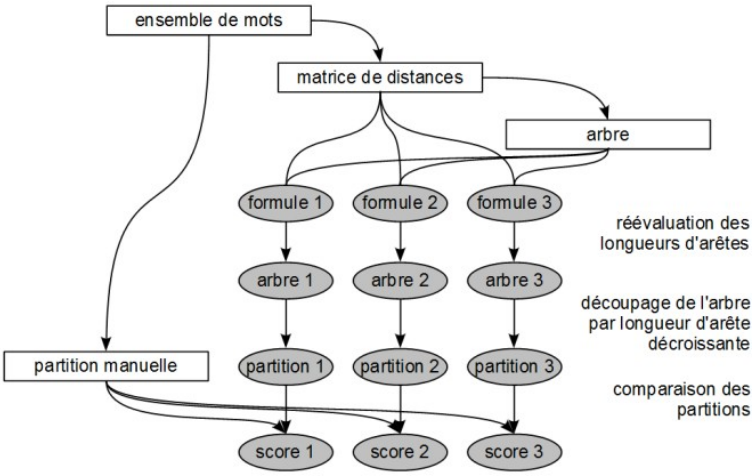


Figure 3 : Méthodologie d'évaluation des formules de calcul de longueur des arêtes de l'arbre.

Plus précisément, pour obtenir un score de qualité, comme il y a *n*-3 arêtes internes dans un arbre de *n* mots, nous sélectionnons parmi ces partitions obtenues après 1, 2, 3... *n*-3 découpages, celle qui est la plus proche de la partition de référence, selon l'indice de Rand, ou selon l'indice de Rand corrigé. Nous obtenons de cette façon deux scores de qualité

(Rand et Rand corrigé) pour chacune des formules de calcul des longueurs d'arêtes de l'arbre.

Rappelons que ces deux indices sont au plus égaux à 1, valeur atteinte pour deux partitions identiques. L'indice de Rand entre deux partitions *P1* et *P2* correspond à la proportion de paires d'éléments qui sont dans la même classe à la fois dans *P1* et dans *P2*, ou dans des classes distinctes à la fois dans *P1* et dans *P2* (Rand, 1971). Comme cet indice a tendance à surestimer la similitude entre deux partitions, l'indice de Rand corrigé (Hubert & Arabie, 1985) a été proposé. Son espérance pour deux partitions aléatoires est nulle, il soustrait donc la part de similitude due au hasard.

3.2 Résultats

Les moyennes des scores de Rand, données en Figure 4, et plus encore celles des scores de Rand corrigé, montrent que pour ces 20 familles de mots, l'utilisation des formules *triples* et *lengthRatio* fournit les meilleures résultats pour fixer les longueurs d'arête.

Formule :	*computed Length*	*triples*	*quartets*	*lengthRatio*	*agreement Pairs*
Rand	0.783	**0.791**	0.762	**0.792**	0.765
Rand corr.	0.354	**0.396**	0.254	**0.392**	0.270

Figure 4 : Moyenne, pour 20 familles de mots de la base Polymots, des scores de Rand et de Rand corrigé pour la meilleure partition construite automatiquement en fonction de la formule choisie pour calculer les longueurs des arêtes.

Pour plus de détails, les scores de Rand et Rand corrigé obtenus pour les 10 premières familles de mots sont donnés en Figures 5 et 6, respectivement.

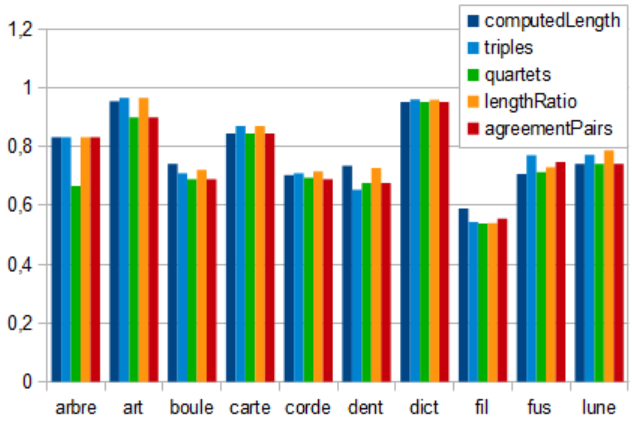


Figure 5 : Score de Rand pour la meilleure partition construite automatiquement en fonction de la formule choisie pour calculer les longueurs des arêtes, pour 10 familles de mots de la base Polymots.

On peut également s'interroger sur la similarité entre ces formules de calcul des distances d'arêtes. Pour en savoir plus, il est possible de comparer les ensembles de longueurs d'arêtes obtenues pour chacune de ces cinq formules. Si l'on se focalise sur l'arbre de la famille du mot “art”, on constate que les longueurs d'arêtes calculées par la formule *lengthRatio* présentent une corrélation avec celles calculées par la formule *triples*, le coefficient de corrélation entre ces ensembles de distances étant de 0,865. Les autres choix de paires de formules ne font en revanche pas apparaître de corrélations aussi nettes. On note également, toujours sur l'arbre de la famille “art”, que les arêtes internes ont une longueur généralement plus importante que les arêtes externes, tant par le calcul avec la formule *triples* que celui avec *lengthRatio*. Ceci est un avantage pour la lisibilité de l'arbre.

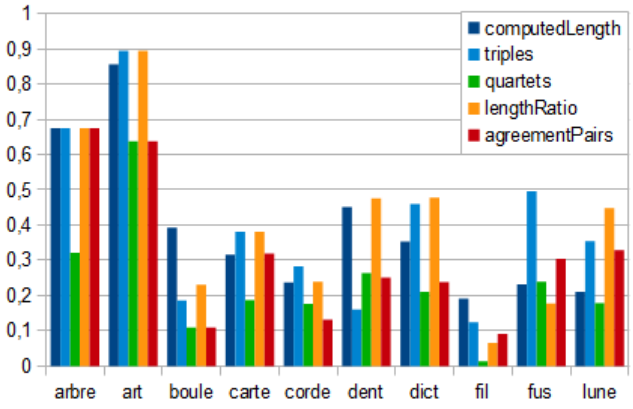


Figure 6 : Score de Rand corrigé pour la meilleure partition construite automatiquement en fonction de la formule choisie pour calculer les longueurs des arêtes, pour 10 familles de mots de la base Polymots.

En revanche, par rapport à la formule *triples*, la formule *lengthRatio* fournit une variance des longueurs d'arêtes moins importante. Pour améliorer la lisibilité, il faudra donc effectuer une transformation monotone des distances (par exemple, une transformation affine) qui augmente cette variance.

4. Évaluation sur un corpus textuel

4.1 Protocole d'évaluation

Dans l'évaluation sur des familles morphologiques ci-dessus, la distance entre les mots de la famille est une composition entre une distance sur les co-occurrents communs au sein du TLFi et une distance des affixes communs. Ainsi, cette distance n'est pas calculée directement à partir d'un corpus textuel, alors que c'est le cas pour les distances de co-occurrence utilisées comme base de la construction des nuages arborés.

Nous proposons donc un second protocole d'évaluation de la qualité de la partition qui se base sur les distances de co-

occurrence entre mots d'un texte implémentées dans TreeCloud. A partir d'une partition d'un ensemble de 25 mots liés à Wikileaks ([julian, assange, porte, parole], [dons, euros, nom, whf], [membres], [wikileaks, site, documents, publication, spécialisé, secrets], [guerre, diplomatie, américaine], [américains, diplomatiques, transparence, monde], [fuites, journaux, publier]), dix textes d'environ 300 mots ont été rédigés. Ce corpus est disponible sur le site treecloud.org.

La consigne suivante a été donnée pour l'élaboration des textes par dix groupes d'étudiants : rédiger "*un texte de plus de 300 mots qui fait obligatoirement apparaître les 25 mots voulus, en tentant de rapprocher les mots contenus dans une même classe de la partition*". La partition de départ provenait d'un découpage thématique en sous-arbres, réalisé manuellement, d'un nuage arboré. Ce nuage arboré montré en Figure 7 a été construit par le logiciel TreeCloud à partir de la concaténation de trois articles de presse : "WikiLeaks : une transparence qui fait débat", dans *Le Monde* du 30 novembre 2010, "WikiLeaks change la donne de la diplomatie et des médias", dans *Les Echos* du 29 novembre 2010, et "Wikileaks, une nébuleuse si peu transparente...", dans *Les Echos* du 9 décembre 2010.

Les 10 textes sont alors concaténés pour former le corpus d'évaluation, et c'est sur ce corpus qu'on applique la méthode de création d'un nuage arboré (fenêtre glissante de 10 mots et pas de glissement de 1 mot pour le calcul des co-occurrences, méthode Neighbor-Joining de Saitou & Nei (1987) pour la construction de l'arbre), puis la méthode indiquée dans la Figure 3 pour le calcul des longueurs d'arêtes, la création de la partition par découpage des arêtes les plus longues, et la comparaison avec la partition de référence, pour chaque formule de calcul des longueurs d'arête. Comme la partition de référence a 7 classes, nous arrêtons le processus de découpage

après 6 découpages d'arêtes, afin de créer la partition dont on évaluera la qualité par rapport à la partition de référence.

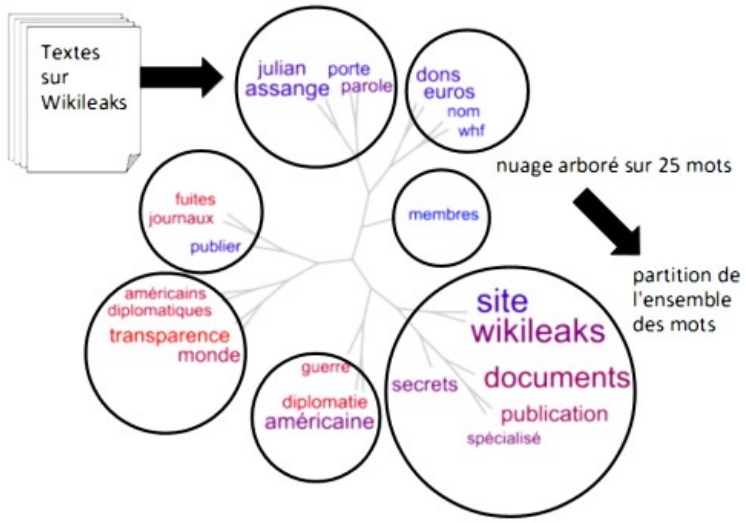


Figure 7 : Nuage arboré de trois articles de presse concaténés permettant de construire une partition de 25 mots liés à Wikileaks (chacune des 7 classes de la partition est contenue dans un cercle) fournie comme base de la rédaction des 10 textes du corpus d'évaluation.

Comme 13 distances de co-occurrence entre mots sont implémentées dans TreeCloud, nous avons choisi de nous focaliser sur les 7 dont la robustesse pour construire des nuages arborés était la meilleure, d'après l'analyse de Gambette & Véronis (2009) : *liddell*, *gmean*, *jaccard*, *dice*, *ms* (minimum sensitivity), *zscore* et *hyperlex* (Gambette, 2010).

4.2 Résultats

Nous obtenons les résultats montrés en Figure 8 pour le score de Rand corrigé. Les formules *lengthRatio* et *triplets* apparaissent encore une fois comme les meilleures, comme on le voit dans la Figure 9 avec les moyennes de score de Rand corrigé.

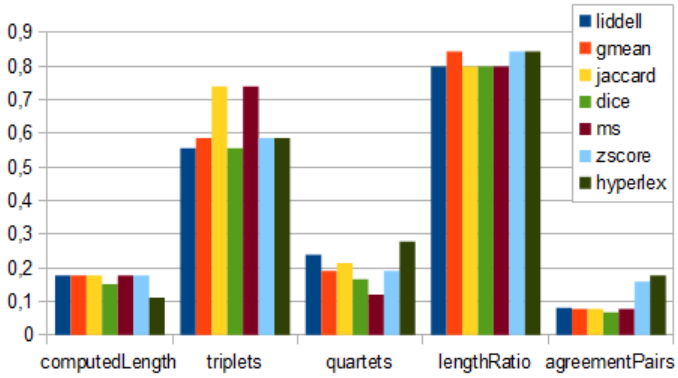


Figure 8 : Score de Rand corrigé des partitions construites automatiquement sur le corpus d'évaluation "Wikileaks", en fonction de 7 distances de co-occurrence et de 5 formules de calcul des longueurs d'arêtes.

Formule :	*computed Length*	*triples*	*quartets*	*lengthRatio*	*agreement Pairs*
Rand corr.	0.164	**0.621**	0.200	**0.818**	0.102

Figure 9 : Moyenne, pour 7 distances de co-occurrence, du score de Rand corrigé des partitions construites automatiquement sur le corpus d'évaluation "Wikileaks", en fonction de 5 formules de calcul des longueurs d'arêtes.

Les partitions obtenues sont très proches de la partition d'origine, voici par exemple celle obtenue avec la formule de co-occurrence *gmean* et la formule de longueurs d'arêtes *lengthRatio*, à comparer avec la partition originale de la Figure 7 : [assange, julian, porte, parole], [site, wikileaks, documents, secrets, publication, spécialisé], [américaine, guerre, diplomatie], [membres], [monde, fuites, diplomatiques, américains, transparence, journaux, publier], [dons, whf, euros, nom].

On peut remarquer que cette partition n'a que 6 classes, car un des 6 découpages n'a pas induit de séparation d'une

classe en deux. En effectuant le découpage de l'arête suivante la plus longue avec ces paramètres (*gmean* et *lengthRatio*), on retrouve exactement la partition originale.

5. Conclusions et perspectives

Cet article fournit une méthodologie de calcul des longueurs d'arêtes d'un arbre de mots qui permet de l'interpréter comme un ensemble de classes, les classes les mieux séparées l'étant par les arêtes les plus longues.

La proposition d'une méthode de partitionnement de l'ensemble des mots aux feuilles de l'arbre par découpage successif des arêtes dans un ordre de longueur décroissant permet de mettre en application ce principe. Comparer les partitions ainsi obtenues à une partition de référence nous a permis de déterminer que deux formules de calcul des longueurs d'arêtes semblent fournir de bons résultats : *triples* (le taux de triplets de mots séparés par cette arête qui le sont également d'après la matrice de distance) et *lengthRatio* (la distance moyenne entre mots de part et d'autre de l'arête, divisée par la distance moyenne entre mots d'un même côté de l'arête).

Ainsi, ces deux formules, ou toute transformation affine (ou plus généralement toute transformation qui respecte l'ordre relatif des arêtes en fonction de leur longueur ainsi calculée), permettent d'obtenir un arbre interprétable comme un ensemble de classes de mots plus ou moins séparées les unes des autres. Ce système augmente la fiabilité de l'interprétation de l'arbre, là où les longueurs directement calculées par la méthode de classification hiérarchique choisie pour construire l'arbre peuvent induire en erreur l'utilisateur qui tente d'interpréter l'arbre.

Comme nous l'avons vu avec la qualité des scores obtenus dans l'évaluation sur un corpus textuel simulant un ensemble d'articles portant sur un même sujet, la méthode de classification non supervisée d'un ensemble de mots en fonction

de leur distance sémantique proposée dans cet article permet d'obtenir de bons résultats, et a donc un intérêt en tant que telle. Une comparaison plus poussée avec d'autres méthodes, et sur d'autres corpus construits de la même manière, permettrait de confirmer la qualité de la méthode.

Remerciements

Nous remercions Alain Guénoche pour les discussions qui sont à l'origine de cet article, et les outils indiqués en réponse à nos besoins méthodologiques. Les remarques du relecteur anonyme de l'article ont également permis de l'améliorer. Le colloque *La co-occurrence : du fait statistique au fait textuel* a également permis de nourrir cet article grâce à plusieurs discussions et présentations. En particulier, les informations données par Jean-Marie Leblanc à propos de sa méthodologie de validation, sur des textes générés à partir du résultat attendu, ont inspiré le second protocole expérimental de cet article. Nous remercions enfin les étudiants du module d'Ingéniérie Linguistique du master 1 mention informatique de l'Université Paris-Est Marne-la-Vallée en 2011-2012, pour leur participation au projet *Classes de mots / Infoling 2012* qui a permis la constitution du corpus utilisé dans ce protocole expérimental, à partir des textes qu'ils ont rédigés.

Références

Dendien J. & Pierrel J.M. (2003). « Le trésor de la langue française informatisé. Un exemple d'informatisation d'un dictionnaire de langue de référence ». *Traitement automatique des langues* 44(2) : 11–37.

Evert S. (2005). *The Statistics of Word Cooccurrences, Word Pairs and Collocations*. Thèse de l'Université de Stuttgart, pp. 75–91.

Felsenstein J. (2004). *Inferring Phylogenies*. Sinauer Associates.

Firth J.R. (1957). « A synopsis of linguistic theory, 1930–1955 ». *Studies in Linguistic Analysis*, pp. 1–32. Special Volume, Philological Society.

Gala N., Hathout N., Nasr A., Rey V. & Seppälä S. (2011). « Création de clusters sémantiques dans des familles morphologiques à partir du TLFi », In *Actes de TALN'11*.

Gala N. & Rey V. (2008). « Polymots : une base de données de constructions dérivationnelles en français à partir de radicaux phonologiques ». In *Actes de TALN'08*.

Gambette P. & Véronis J. (2009). « Visualising a Text with a Tree Cloud ». In Locarek-Junge H. and Weihs C., éditeurs, *Classification as a Tool of Research, Proc. of IFCS'09*, pp. 561–570.

Gambette P. (2010). « User manual for TreeCloud ». Manuscrit, http://manual.treecloud.com.

Guénoche A. & Garreta H. (2002). « Representation and Evaluation of Partitions ». In *Classification, clustering and data analysis, Proc. of IFCS'02*.

Hubert L. & Arabie P. (1971), « Comparing Partitions », *Journal of Classification* 2(1) : 193–218.

Huson D.H. & Bryant D. (2006). « Application of Phylogenetic Networks in Evolutionary Studies ». *Molecular Biology and Evolution* 23(2) : 254–267, logiciel disponible sur www.splitstree.org.

Luong X. (1989). *Analyse arborée des données textuelles*. CUMFID 16.

Mayaffre D. (2008). « Quand “travail”, “famille”, “patrie” co-occurrent dans le discours de Nicolas Sarkozy. Étude de cas et réflexion théorique sur la co-occurrence ». In Heiden

S., Pincemin B., éditeurs, *Actes des JADT'08*, pp. 811–822.

Rand W.M. (1971), « Objective criteria for the evaluation of clustering methods », *Journal of the American Statistical Association* 66(336) : 846–850.

Saitou N. & Nei M. (1987), « The neighbor-joining method: a new method for reconstructing phylogenetic trees », *Molecular Biology and Evolution* 4(4) : 406–425.

Locating a Tree in a Phylogenetic Network in Quadratic Time

Philippe Gambette, Andreas D.M. Gunawan, Anthony Labarre, Stéphane Vialette, Louxin Zhang

Locating a Tree in a Phylogenetic Network in Quadratic Time

Philippe Gambette[1], Andreas D. M. Gunawan[2], Anthony Labarre[1], Stéphane Vialette[1], and Louxin Zhang[2]

[1] Université Paris-Est, LIGM (UMR 8049), UPEM, CNRS, ESIEE, ENPC, F-77454, Marne-la-Vallée, France
[2] Department of Mathematics, National University of Singapore

Abstract. A fundamental problem in the study of phylogenetic networks is to determine whether or not a given phylogenetic network contains a given phylogenetic tree. We develop a quadratic-time algorithm for this problem for binary nearly-stable phylogenetic networks. We also show that the number of reticulations in a reticulation visible or nearly stable phylogenetic network is bounded from above by a function linear in the number of taxa.

1 Introduction

Genetic material can be transferred between organisms by hybridization, recombination and horizontal gene transfer besides traditional reproduction. Recent studies in comparative genomics suggest that these "lateral" processes are a driving force in evolution which shapes the genome of a species [3, 11, 16]. Accordingly, phylogenetic networks have commonly been used to model reticulate evolutionary histories of species [3, 4, 10]. A plethora of methods for reconstructing reticulate evolutionary histories of species and related algorithmic issues have extensively been studied over the past two decades [5, 6, 12–14, 17].

A *phylogenetic network* is an acyclic digraph with a set X of labeled leaves (that is, vertices of outdegree zero) and a root (having indegree zero). The leaves are in one-to-one correspondence with a collection of taxa under study, whereas the unique root represents their least common ancestor. Vertices with indegree one represent speciation events. Vertices of indegree at least two represent an evolutionary process by which genetic material was horizontally transferred from one species to another.

A fundamental question in the study of phylogenetic networks is to determine whether a tree is *displayed* by a phylogenetic network over the same set of taxa (in a sense we define precisely below). This problem is called the *tree containment problem* [6]. Answering this question is indeed useful to validate and justify a phylogenetic network model by testing whether it displays existing phylogenies over a set of taxa under study.

The problem is NP-complete in general [9], even on the more restricted class of *tree-sibling time-consistent regular networks* [7]. Although great effort has been

devoted to the study of that problem, it has been shown to be polynomial-time solvable only for a couple of interesting classes of phylogenetic networks, namely, *normal* networks and *tree-child* networks [7]. Determining the complexity of the tree containment problem for a class of phylogenetic networks that properly contains tree-child networks, particularly those with the so-called *reticulation-visibility property*, is an open problem [6, 7].

In this paper, we study the tree containment problem for *nearly stable* phylogenetic networks (defined in the next section), which generalize normal and tree-child networks. Recombination histories of viruses, hybridization histories of plants, and histories of horizontal gene transfers reported in literature often satisfy the property that defines those networks [8, 10]. Our key results include: (i) the number of reticulations in a reticulation-visible or nearly stable phylogenetic network is linearly bounded from above in terms of the number of taxa; and (ii) the tree containment problem for nearly stable phylogenetic networks can be solved in quadratic time. Omitted proofs and details will appear in the extended version.

2 Concepts and Notions

A *(phylogenetic) network* on a set X of taxa is a directed acyclic graph with a single root (a vertex with indegree 0) which satisfies the following properties: (i) its leaves (vertices with outdegree 0) are in one-to-one correspondence with the taxa in X; (ii) there are no vertices with both indegree one and outdegree one; and (iii) there is a path from the root to any other vertex. We identify each leaf with the taxon corresponding to it and refer to the directed edges (tail, head) as *branches*.

In a network, *reticulation vertices* (or simply *reticulations*) are vertices with indegree at least two and outdegree one; *tree vertices* are vertices with indegree one and outdegree at least two. A branch is a *tree branch* if it ends at a tree vertex; it is called a *reticulation branch* otherwise.

A network is *binary* if its root, leaves and the other vertices have degree 2, 1 and 3, respectively. A *phylogenetic tree* is simply a binary network without reticulations.

For a binary network N, we shall use r_N to denote the root of N. Let x and y be vertices in N. We say that x is a *parent* of y and y is a *child* of x if (x, y) is a branch. More generally, we say that x is an *ancestor* of y and equivalently y is a *descendant* of x if there is a directed path from x to y. A vertex x in N is a *stable ancestor* of a vertex v if it belongs to all directed paths from r_N to v. We say that x is *stable* if there exists a leaf ℓ such that x is a stable ancestor of ℓ.

Proposition 1. *Let N be a binary network. The following facts hold.*

(1) *A vertex is stable if it has a stable tree child.*
(2) *A reticulation is stable if and only if its unique child is a stable tree vertex.*
(3) *If a tree vertex is stable, then its children cannot both be reticulations.*

A network is a *tree-child* network if every vertex has a child that is a tree vertex [2]. It can be proved that a network is a tree-child network if and only if every vertex is stable. It is *reticulation-visible* if all its reticulations are stable [6]. It is *nearly stable* if for every vertex, either that vertex is stable or its parents are.

Contracting a branch (u, v) means replacing it with a single vertex w in such a way that all neighbors of u and v become neighbors of w. Given a binary phylogenetic tree T and a binary network N, we say that N *displays* T if there is a spanning subtree T' of N that is a *subdivision* of T, i.e. T' has the same vertex set as N and T can be obtained from T' by contracting all branches in T' incident with the vertices with outdegree 1 and indegree 1, all branches incident with the "dummy leaves" (leaves in T' that correspond to tree vertices in N), and all branches incident with a vertex of indegree 0 and outdegree 1. Figure 1 shows an example of a phylogenetic network N and a tree that is displayed in N.

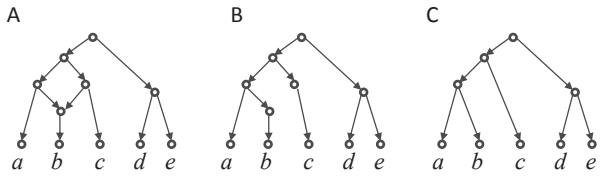


Fig. 1. (A) A phylogenetic network. (B) A spanning subtree of N obtained after the reticulation branch between the parents of c and b is removed. (C) A tree displayed in N through the subtree in (B).

In this work, we study the *tree containment problem* (TCP), which is that of determining whether a phylogenetic tree is displayed by a network or not.

3 How Many Reticulations in a Network?

An arbitrary network with n leaves can have a very large number of reticulations. To analyze the time complexity of an algorithm designed for solving a network problem, we need to bound the size of the network by a function of n.

Removing a reticulation branch from each reticulation in a binary network N yields a spanning subtree T'. All leaves in N are still leaves in T', but T' may additionally contain some "dummy leaves" that correspond to tree vertices whose outgoing branches have both been removed. The following lemma says that it is always possible to remove proper reticulation branches so as to obtain a tree without dummy leaves.

Lemma 1. *Let N be a binary reticulation-visible phylogenetic network. We can determine which reticulation branch to remove at each reticulation so that the tree obtained after removing the selected branches contains no dummy leaves.*

Proof. Let T be a tree obtained from N by removing exactly one reticulation branch incident to each reticulation. In order for T not to contain any dummy leaves, we need to guarantee that the reticulation branches to be removed are incident with different tree vertices. In other words, the branches to be removed form a matching that covers every reticulation in N. Since N has the reticulation-visibility property, the parents of each reticulation are both tree vertices (Proposition 1). Such a set of reticulation branches exists and can be found by applying Hall's Theorem to a bipartite graph with tree vertices and reticulations as vertex sets and reticulation branches as edges. Since each reticulation is the head of two reticulation branches and each tree vertex is the tail of at most two reticulation branches, there exists a matching that covers all the reticulations (see a result of N. Alon on page 429 in [1]). □

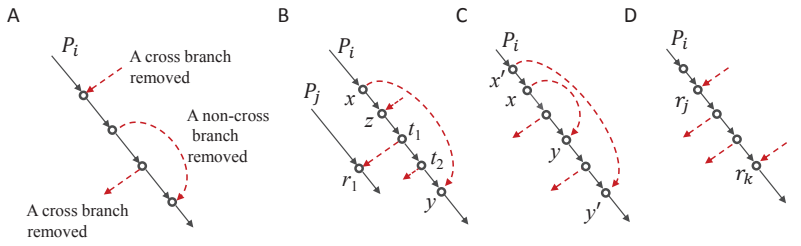


Fig. 2. Illustration of the different cases in the proof of Theorem 1. **A**. Definition of cross and non-cross branches removed from a path. **B**. The branch (x, y) is a non-cross branch removed from a path. Assume that a cross branch (z', z) has been removed from a reticulation z inside the segment from x and y, where z' is not shown, and two cross branches have also been removed from two tree vertices t_1 and t_2 between z and y. **C**. Some cross branches must have been removed from their tails located between the heads of two non-cross branches that are removed from a path (in this case, between y and y'). **D**. If two cross branches have been removed from two reticulations in a path, then the upper reticulation (r_j here) is not stable.

Theorem 1. *Let N be a binary reticulation-visible phylogenetic network with n leaves. Then N has at most $4(n-1)$ reticulations.*

Proof. Assume N contains m reticulations. By Lemma 1, we can obtain a tree T without dummy leaves by removing m reticulation branches from N. Since N is binary, an internal vertex in T has either one or two children; equivalently, T is a subdivision of a rooted binary tree T' over the same leaves as N. Therefore, T' has $n-1$ internal vertices (including its root) of outdegree 2 and there are $2n-2$ paths P_i $(1 \leq i \leq 2n-2)$ satisfying (i) the ends of each P_i are either the root of T, a leaf, or internal vertices of outdegree 2, and (ii) each internal vertex of P_i has both indegree and outdegree 1 if P_i consists of two or more branches.

For each path P_i of length ≥ 2, an internal vertex of P_i is either a tree vertex of N, whose outgoing branch not in P_i has been removed, or a reticulation, whose incoming branch not in P_i has been removed. For convenience of discussion, we divide the removed reticulation branches into **cross** and **non-cross** branches (with respect to T) (Figure 2A). A removed branch is called a *cross branch* if its tail and head are located on two different paths P_i and P_j, $i \neq j$, otherwise it's called a *non-cross branch*. We first have the following facts.

Facts

(1) *If (x, y) is a non-cross branch removed from P_i, then at least one cross branch has been removed from its tree vertex tail in the segment $P_i[x, y]$ from x to y of P_i, and there is no reticulation in $P_i[x, y]$ other than y.*

(2) *Let (x, y) and (x', y') be two non-cross branches removed from P_i, where y is an ancestor of y'. Then there exists at least one cross branch being removed from its tree vertex tail located between y and y' (*Figure 2C*).*

(3) *There are at least as many cross reticulation branches removed as non-cross reticulation branches.*

Proof. (1) Since N contains no parallel branches, $P_i[x, y]$ has at least three vertices, so it suffices to prove that y is the only reticulation in $P_i[x, y]$.
Assume on the contrary that a branch (z', z) has been removed from a reticulation z in $P_i[x, y]$ (Figure 2B). Then there is a path including (x, y) from r_N to a leaf below y that avoids z, so z is not stable on any leaf below y (and hence below z) in T (and hence in N). Moreover, since T is a subtree of N, z cannot be stable in N on any leaf that is not below z in T. N and T have the same leaf set, hence z is not stable in N, contradicting the reticulation-visibility property.
(2) Note that y and y' are reticulations in N. By Fact (1) above, y must be above x', and there is a cross brach removed from its tree vertex tail located between x' and y'.
(3) By Facts (1) and (2), we can establish an injective map from the set of non-cross reticulation branches to that of cross ones. Hence, the statement in this part is also true. □

Assume at least $2n-1$ cross branches (t_i, r_i) have been removed from the $2n-2$ paths P_i. At least two heads r_j and r_k are on the same path P_i (Figure 2D). Using an argument similar to that used in the proof of Fact (2), one of r_j and r_k which is upstream in P_i is not stable, a contradiction. Therefore, at most $2n-2$ cross branches have been removed to produce T. By Fact (3), there are also at most $2n-2$ non-cross branches removed during the process. Since we removed one incoming branch for each reticulation, we conclude that there are at most $4(n-1)$ reticulations in N. □

Lemma 2. *Let N be a binary nearly stable network, and let $U_{\text{ret}}(N)$ (resp. $S_{\text{ret}}(N)$) denote the number of all unstable (resp. stable) reticulations in N. We can transform N into a binary reticulation-visible network N' with the property that N' has the same leaf set as N and $S_{ret}(N) \leq S_{ret}(N') \leq S_{ret}(N)+U_{ret}(N)$.*

Proof. Let a be an unstable reticulation in N, whose child is denoted by b. Since N is nearly stable, b is stable. By Proposition 1(2), b is a stable reticulation. Let c denote a parent of a; then c is stable by definition of N, and it is a tree vertex by Proposition 1(2). Let d denote the other child of c. Since c is stable, d is a tree vertex (Proposition 1(3)). In addition, d is stable.

Assume on the contrary that d is unstable. Then both its children must be stable by the nearly-stable property of N. Hence, by Proposition 1(2) and the fact that d is unstable, both its children are stable reticulations. Since a is unstable, a is not a child of d. This implies that c is unstable, a contradiction.

Finally, let e be the parent of c. f be the other parent of a and g be the other parent of b (see Figure 3). Note that $g \neq f$. Otherwise, f is unstable, contradicting that there are no two consecutive unstable vertices. To transform N into a binary reticulation-visible network, we remove unstable vertex a by first removing the branch (c, a), and then contracting the paths f-a-b and e-c-d into branches (f, b) and (e, d). Both b and d are clearly still stable in the resulting network. By rewiring around every unstable reticulation in N, we produce a binary reticulation-visible network N'. The inequality follows from the fact that no stable reticulation is removed, and no new reticulation is created during the rewiring. □

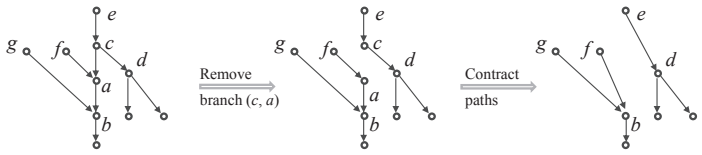


Fig. 3. A An unstable reticulation a, its stable child b and its stable parents (c and f) in the original network N. To transform N into a reticulation-visible network, we remove the incoming reticulation branch (c, a) (**B**) and then contract paths e-c-d and f-a-b (**C**). The rewiring eliminates the unstable reticulation vertex a.

Lemma 3. *For a binary nearly stable network N, $U_{ret}(N) \leq 2S_{ret}(N)$.*

Proof. Directly follows from the fact that an unstable reticulation must have a stable reticulation as its child, and any stable reticulation can be the child of at most two unstable reticulations. □

Theorem 2. *Let N be a binary nearly stable network with n leaves. Let $T(N)$ denotes the number of tree vertices in N. Then:*

(i) N has at most $12(n-1)$ reticulations;
(ii) $|T(N)| \leq 13(n-1)$ and $|E(N)| \leq 38(n-1)$.

Proof. (i) Theorem 1 and Lemmas 2 and 3 imply $S_{ret}(N)+U_{ret}(N) \leq 3S_{ret}(N) \leq 3S_{ret}(N') \leq 3(4n-4) = 12(n-1)$.
(ii) We can think of the network as a flow network, with r_N as source and the n leaves as sinks. Hence, the number of tree vertices equals $n-1$ plus the number of reticulations, that is, at most $13(n-1)$ (by (i)). Since the outdegree of the root is two, and the outdegrees of each tree and reticulation vertex are 2 and 1, respectively, N has $2(13n-13)+12(n-1) = 38(n-1)$ branches at most. □

4 A Quadratic-Time Algorithm for the TCP

In this section, we shall present a quadratic-time algorithm for solving the TCP. If a given network N and a given reference tree T contain a common subphylogeny, then we can simplify the task of determining whether N displays T by replacing the common subphylogeny by a new leaf. Therefore, without loss of generality, we assume that N does not contain a subphylogeny with two or more leaves. We call this property the *subphylogeny-free property.*

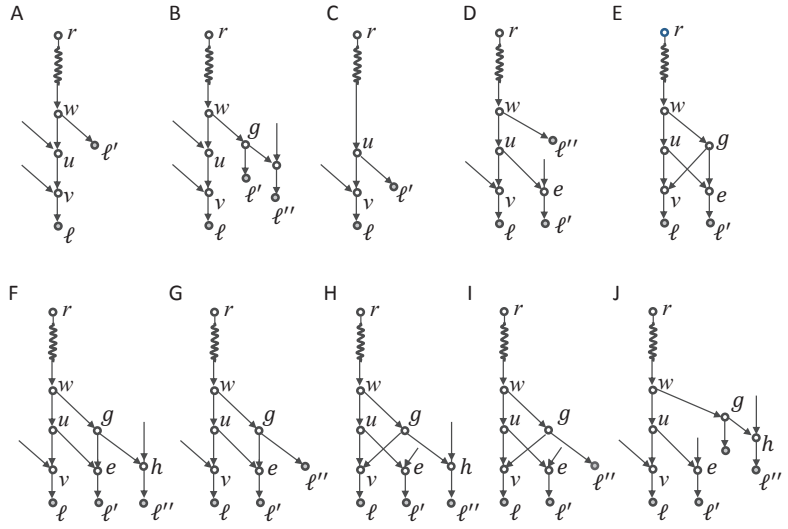


Fig. 4. All ten possible subnetworks at the end of a longest path in a nearly stable network. Here, r is the network root and the directed path from r to w is represented by a coiled path. The parent w of u is not shown in **C**.

Lemma 4. *Let N be a nearly stable phylogenetic network satisfying the subphylogeny-free property. Let $P = (r, \ldots, w, u, v, \ell)$ be a longest root-to-leaf path of four or*

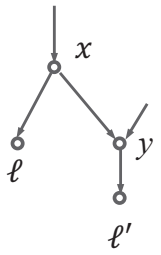
more vertices in N, where $r = r_N$ and ℓ the leaf end. Then the subnetwork consisting of the descendants of w exhibits one of the structures given in Figure 4.

Proof. Note that v cannot be a tree vertex: since P is a longest root-to-leaf path, the other child of v would otherwise be a leaf, thereby contradicting our assumption that N satisfies the subphylogeny-free property. Therefore, v is a reticulation. There are two possible cases for u.

1. **The u is a reticulation:** Then u is unstable, and w must be a stable tree vertex (see Proposition 1(2) for both claims), which is stable on ℓ or some other leaf. Let g be the other child of w. By Proposition 1(3), g is either a tree vertex or a leaf. If g is a leaf, we obtain the subnetwork in Figure 4A. If g is a tree vertex, then neither of its children is a tree vertex: since P is a longest path, a tree vertex child of g would have two leaves as children, thereby contradicting the subphylogeny-free property. Note that g's children cannot both be reticulations either, since otherwise w would be unstable. Therefore, one child of g is a leaf and the other is a reticulation with a leaf child (again because P is a longest path), as shown in Figure 4B.
2. **The u is a tree vertex:** Let e denote the other child of u. Note that e cannot be a tree vertex, otherwise both its children would be leaves (since P is a longest path), which would contradict our assumption that N has the subphylogeny-free property. If e is a leaf, we obtain the subnetwork shown in Figure 4C. If e is a reticulation, then its only child is a leaf (again because P is a longest path), so e is stable on that leaf and u is therefore unstable. Since N is nearly stable, w must be a stable tree vertex. We consider the other child g of w in the following subcases.
   (2.1) If g is a leaf, then we have the subnetwork given in Figure 4D.
   (2.2) If g is a tree vertex and also a parent of e and v, then we obtain the subnetwork in Figure 4E.
   (2.3) If g is a tree vertex and in addition, g is a parent of e, but not a parent of v: then w is stable on ℓ', the unique child of e. Let h be the other child of g; then h cannot be a tree vertex, since both its children would then be leaves, which would contradict our assumption that N has the subphylogeny-free property. If h is a reticulation, its child must be a leaf, since P is a longest path. Thus, we have the subnetwork given in Figure 4F. If h is a leaf, we obtain the subnetwork in Figure 4G.
   (2.4) If g is a tree vertex and in addition, g is a parent of v, but not a parent of e, then a discussion similar to that of case (2.2) characterises the only two possible subnetworks (Figure 4H and 4I) in this case.
   (2.5) If g is a tree vertex and in addition, g is neither a parent of v nor a parent of e: then again we look at g's children. Both cannot be reticulations, otherwise w is unstable, a contradiction. If neither of them is a reticulation, then there is a subtree below g; if one of them is a reticulation and the other is a tree vertex, then again there is a subtree. The only possible case that remains, shown in Figure 4J, is the case where one child is a reticulation and the other is a leaf.

(2.6) If g is a reticulation: Then w unstable. This is impossible, as w is a stable tree vertex. □

The subnetwork below g of the structures shown in Figure 4B, 4G, 4I, 4J and that below u in Figure 4C match the following pattern:



in which a leaf ℓ has a reticulation sibling y and a leaf nephew, ℓ'. Such a pattern is called an *uncle-nephew structure*. Note that if ℓ and ℓ' are not siblings in a tree displayed by N, then the reticulation branch (x, y) should not be used. If ℓ and ℓ' are siblings, either (x, y) or the other branch entering y can be used. Here, since the other branch enters y from an unspecified vertex, it is simply called a *dangling branch*. It is not hard to see that for a tree T in which ℓ and ℓ' are siblings, if T is displayed in the network resulting from the removal of (x, y), it is also displayed in the one after the dangling branch is removed. Hence, to determine whether N displays a tree T, we can simplify the network by eliminating y using the following process:

> **Uncle-Nephew Reduction** In an uncle-nephew substructure shown above, remove the dangling branch if ℓ and ℓ' are siblings in T, or remove (x, y) otherwise. Then contract vertices with indegree and outdegree 1.

In each of the other cases, we can also simplify the network by using information on the input tree. To summarize how to simplify the network, we use the following notation for each vertex w in a network N:

- $R(w)$ denotes the subnetwork consisting of all the descendants of w;
- $(-, x)$ denotes the dangling branch entering x from its parent not in $R(w)$ for x in $R(w)$;
- $N' + (x, y)$ denotes the subnetwork obtained by adding (x, y) into N' for a subnetwork N' of N and a branch (x, y) of N;
- $N' - (x, y)$ denotes the subnetwork obtained by removing (x, y) from N' for a subnetwork N' of N;
- $p_T(x)$ denotes the parent of a vertex x in a tree T.

Theorem 3. *Let N be a binary nearly stable network with no uncle-nephew structure, and T a tree with the same set of labeled leaves. Let w be a tree vertex in N. Define N' as follows.*

(i) *When $R(w)$ matches the structure of Figure 4A, define $N' = N - (w, u)$ if ℓ and ℓ' are not sibling in T and $N' = N - \{(-, u), (-, v)\}$ otherwise.*

(ii) *When $R(w)$ matches the structure of Figure 4D, define $N' = N - (-, v)$ when ℓ and ℓ'' are siblings, or when ℓ and ℓ' are siblings and their parent is a sibling of ℓ'' in T, and $N' = N - (u, v)$ otherwise.*

(iii) *When $R(w)$ matches the structure of Figure 4E, define $N' = N - \{(u, e), (g, v)\}$.*

(iv) *When $R(w)$ matches the structure of Figure 4F, define $N' = N - \{(g, e), (-, v)\}$ if ℓ and ℓ' are siblings in T and $N' = N - (u, e)$ otherwise.*

(v) *When $R(w)$ matches the structure of Figure 4H, define $N' = N - \{(g, v), (-, e)\}$ if ℓ and ℓ' are siblings in T and $N' = N - (u, v)$ otherwise.*

Then N' is nearly stable and N displays T only if N' displays T.

Proof. Since none of the simplifications removes any leaf and all of them only reduce possible paths from r_N to a leaf, the resulting network N' is nearly stable.

Assume $R(w)$ is the subnetwork in Figure 4A and N displays T. Then there exists a subtree T' of N that is a subdivision of T and let $p_T(\ell)$ corresponds x in T'. Clearly, x is of degree 3 and hence a tree vertex in N. We consider two cases.

CASE A. Leaves ℓ and ℓ' are not siblings in T.

We first have that $x \neq u$, $x \neq v$ for u and v in Figure 4A. We also have $x \neq w$. Otherwise, ℓ' must be a child of x in T' and ℓ is a sibling of ℓ' in T, a contradiction. Therefore, the path from x to ℓ contains two or more vertices and v is the parent of ℓ in this path. If u is the parent of v in the same path, neither $(-, v)$ nor (w, u) is in T', indicating that $N' = N - (w, u)$ also displays T.

If $p_{T'}(v) \neq u$ in the same path, then (u, v) is not in T' and hence u becomes a dummy leaf in T', as there is no leaf other than ℓ below u in $R(w)$. If (w, u) is in T', then $(-, u)$ is not in T' and $T' + (-, u) - (w, u)$ is a subtree of N' in which only the dummy leaf u is relocated. Hence, N' also displays T.

CASE B. Leaves ℓ and ℓ' are siblings in T.

Then x is a common ancestor of ℓ and ℓ' in N. If $x = w$, the path from x to ℓ in T' must be w, u, v, as this is only path from w to ℓ in N. Hence, $(-, u)$ and $(-, v)$ are not in T'. Therefore, T' is a subtree of N' and N' also displays T.

If $x \neq w$, then x is an ancestor of w and hence w is the parent of ℓ' in the path from x to ℓ' in T'. Note that $p_{T'}(\ell) = v$. If $p_{T'}(v) = u$, then $(-, u)$ is in T', but both $(-, v)$ and (w, u) are not. $T'' = T' + (w, u) - (-, u)$ is a subtree of N'. Noting that T'' is also a subdivision of T, N' displays T.

If $p_{T'}(v) \neq u$, then $(-, v)$ is in the path from x to ℓ in T'. This implies that (u, v) is not in T' and u is a dead-end in T'. If (w, u) is in T', the subtree $T'' = T' + (u, v) - (-, v)$ of N' is a subdivision of T. If (w, u) is not in T', the subtree $T'' = T' + (w, u) - (-, u) - (-, v)$ of N' is a subtree of N'. Hence, N' displays T'.

Similarly, we can prove that N displays T only if N' displays T when $R(w)$ is the subnetwork in the panels D, F, and H in Figure 4. Note also that the subnetworks in the panels F and H are essentially identical (if the positions of v and e are switched). Due to the limited space, the details are omitted here. The case when $R(w)$ is the subnetwork in Figure 4E is trivial, as deletion of which two reticulation branches from v and e does not affect outcome. □

By Theorem 3, we are able to determine whether a nearly stable phylogenetic network N displays a binary tree T or not by repeatedly executing the following tasks in turn until the resulting network N' becomes a tree:

- Compute a longest path P in $N' = N$;
- Simplify N' by considering the subnetwork at the end of P according to the cases in Lemma 4;
- Contract degenerated reticulations in N' and replace the parent of a pair of leaves appearing in both N' and T with a new leaf.

and then check if N' is identical to T.

Finally, we analyze the time complexity. Let N and T have n leaves. By Theorem 2, there are $O(n)$ vertices and $O(n)$ branches in N. Since we eliminate at least a reticulation in each loop step, the algorithm stops after $O(n)$ loop steps. In each loop step, a longest path can be computed in $O(n)$ time ([15], page 661), as N is acyclic; both the second and third tasks can be done in constant time. In summary, our algorithm has quadratic time complexity.

5 Conclusion

We have developed a quadratic-time algorithm for the TCP for binary nearly stable phylogenetic networks. Our algorithm not only is applicable to a superclass of tree-child networks, but also has a lower time complexity than the algorithm reported in [7]. Although phylogenetic network models built in the study of viral and plant evolution are often nearly stable, it is interesting to know whether the TCP is polynomial time solvable or not for networks with other weak properties.

In particular, the problem remains open for binary networks with the visibility property, but the upper bound we have presented on the number of reticulation vertices of such networks, as well as our algorithm for nearly stable phylogenetic networks, provide definitely valuable ideas to solve the problem, exactly or heuristically, on phylogenetic networks with the reticulation visibility property.

6 Acknowledgments

The project was financially supported by Merlion Programme 2013.

References

1. Bondy, J.A., Murty, U.S.R.: Graph Theory. Springer (2008)
2. Cardona, G., Rosselló, F., Valiente, G.: Comparison of tree-child phylogenetic networks. IEEE/ACM Trans. Comput. Biol. Bioinfo. 6(4), 552–569 (2009)
3. Chan, J.M., Carlsson, G., Rabadan, R.: Topology of viral evolution. PNAS 110(46), 18566–18571 (2013)

4. Dagan, T., Artzy-Randrup, Y., Martin, W.: Modular networks and cumulative impact of lateral transfer in prokaryote genome evolution. PNAS 105(29), 10039–10044 (2008)
5. Gusfield, D.: ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks. The MIT Press (2014)
6. Huson, D.H., Rupp, R., Scornavacca, C.: Phylogenetic Networks: Concepts, Algorithms and Applications. Cambridge University Press (2011)
7. van Iersel, L., Semple, C., Steel, M.: Locating a tree in a phylogenetic network. Inf. Process. Lett. 110(23), 1037–1043 (2010)
8. Jenkins, P., Song, Y., Brem, R.: Genealogy-based methods for inference of historical recombination and gene flow and their application in *saccharomyces cerevisiae*. PLoS ONE 7(11), e46947 (2012)
9. Kanj, I.A., Nakhleh, L., Than, C., Xia, G.: Seeing the trees and their branches in the network is hard. Theor. Comput. Sci. 401, 153–164 (2008)
10. Marcussen, T., Jakobsen, K.S., Danihelka, J., Ballard, H.E., Blaxland, K., Brysting, A.K., Oxelman, B.: Inferring species networks from gene trees in high-polyploid north american and hawaiian violets (*viola*, violaceae). Syst. Biol. 61, 107–126 (2012)
11. McBreen, K., Lockhart, P.J.: Reconstructing reticulate evolutionary histories of plants. Trends Plant Sci. 11(8), 103–122 (2006)
12. Moret, B.M.E., Nakhleh, L., Warnow, T., Linder, C.R., Tholse, A., Padolina, A., Sun, J., Timme, R.: Phylogenetic networks: Modeling, reconstructibility, and accuracy. IEEE/ACM Trans. Comput. Biol. Bioinfo. 1(1), 13–23 (2004)
13. Nakhleh, L.: Computational approaches to species phylogeny inference and gene tree reconciliation. Trends Ecol. Evolut. 28(12), 719–728 (2013)
14. Parida, L. : Ancestral recombinations graph: a reconstructability perspective using random-graphs framework. J. Comput. Biol. 17(10), 1345–1370 (2010)
15. Sedgewick, R., Wayne, K.: Algorithms, 4th Edition. Addison-Wesley (2011)
16. Treangen, T.J., Rocha, E.P.: Horizontal transfer, not duplication, drives the expansion of protein families in prokaryotes. PLoS Genetics 7(1), e1001284 (2011)
17. Wang, L., Zhang, K., Zhang, L.: Perfect phylogenetic networks with recombination. J. Comp. Biol. 8(1), 69–78 (2001)

Do branch lengths help to locate a tree in a phylogenetic network?

Philippe Gambette, Leo van Iersel, Steven Kelk, Fabio Pardi, Celine Scornavacca

Noname manuscript No.
(will be inserted by the editor)

Do branch lengths help to locate a tree in a phylogenetic network?

Philippe Gambette · Leo van Iersel · Steven Kelk · Fabio Pardi · Celine Scornavacca

Received: date / Accepted: date

P. Gambette
Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-77454, Marne-la-Vallée, France
E-mail: philippe.gambette@u-pem.fr

L. van Iersel
Delft Institute of Applied Mathematics, Delft University of Technology
Postbus 5031,2600 GA Delft, The Netherlands
E-mail: l.j.j.v.iersel@gmail.com

S. Kelk
Department of Data Science and Knowledge Engineering (DKE)
Maastricht University, P.O. Box 616, 6200 MD, Maastricht, The Netherlands
E-mail: steven.kelk@maastrichtuniversity.nl

Corresponding author: F. Pardi
Institut de Biologie Computationnelle (IBC)
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)
CNRS, Université de Montpellier, France
E-mail: pardi@lirmm.fr

C. Scornavacca
Institut de Biologie Computationnelle (IBC)
Institut des Sciences de l'Evolution, CC 064
Place Eugène Bataillon, Montpellier, France
E-mail: celine.scornavacca@umontpellier.fr

Abstract Phylogenetic networks are increasingly used in evolutionary biology to represent the history of species that have undergone reticulate events such as horizontal gene transfer, hybrid speciation and recombination. One of the most fundamental questions that arise in this context is whether the evolution of a gene with one copy in all species can be explained by a given network. In mathematical terms, this is often translated in the following way: is a given phylogenetic tree contained in a given phylogenetic network? Recently this *tree containment* problem has been widely investigated from a computational perspective, but most studies have only focused on the topology of the phylogenies, ignoring a piece of information that, in the case of phylogenetic trees, is routinely inferred by evolutionary analyses: branch lengths. These measure the amount of change (e.g., nucleotide substitutions) that has occurred along each branch of the phylogeny. Here, we study a number of versions of the tree containment problem that explicitly account for branch lengths. We show that, although length information has the potential to locate more precisely a tree within a network, the problem is computationally hard in its most general form. On a positive note, for a number of special cases of biological relevance, we provide algorithms that solve this problem efficiently. This includes the case of networks of limited complexity, for which it is possible to recover, among the trees contained by the network with the same topology as the input tree, the closest one in terms of branch lengths.

Keywords Phylogenetic network · tree containment · branch lengths · displayed trees · computational complexity

1 Introduction

The last few years have witnessed a growing appreciation of reticulate evolution – that is, cases where the history of a set of taxa (e.g., species, populations or genomes) cannot be accurately represented as a phylogenetic tree [11,2], because of events causing inheritance from more than one ancestor. Classic examples of such reticulate events are hybrid speciation [29,32,1], horizontal gene transfer [5,19,40] and recombination [34,36]. Inferring the occurrence of these events in the past is a crucial step towards tackling major biological issues, for example to understand recombinant aspects of viruses such as HIV [35], or characterizing the mosaic structure of plant genomes.

Reticulate evolution is naturally represented by *phylogenetic networks* – mathematically, simple generalizations of phylogenetic trees, where some nodes are allowed to have multiple direct ancestors [21,31]. Currently, much of the mathematical and computational literature on this subject focuses solely on the topology of phylogenetic networks [22], namely not taking into account branch length information. This information – a measure of elapsed time, or of change that a species or gene has undergone along a branch – is usually estimated when inferring phylogenetic trees, and it may have a big impact on the study of reticulate evolution as well.

For example, in the literature investigating hybridization in the presence of incomplete lineage sorting, the branch lengths of a phylogenetic network are the key parameters to calculate the probability of observing a gene tree, and thus to determine the likelihood of the network [30,38]. Moreover, accurate estimates of branch lengths in the gene trees are known to improve the accuracy of the inferred network [28,39]. Similarly, for another large class of methods for network reconstruction, otherwise indistinguishable network scenarios can become distinguishable, if branch lengths are taken into account [33]. The precise meaning of branch lengths is often context-dependent, ranging from expected number of substitutions per site, generally adopted in molecular phylogenetics, to a measure of the probability of coalescence, often adopted for smaller timescales where incomplete lineage sorting is common, to the amount of time elapsed. In the last case, we may expect the phylogeny (network or tree) to be ultrametric, that is to have all its leaves at the same distance from the root [7,4].

In this paper, we explore the impact of branch lengths on a fundamental question about phylogenetic networks: the *tree containment* problem. Informally (formal definitions will be given in the next section), this problem involves determining whether a given phylogenetic tree is contained, or *displayed*, by a given phylogenetic network, and in the positive case, locating this tree within the network. Biologically, this means understanding whether a gene – whose phylogenetic history is well-known – is consistent with a given phylogenetic network, and understanding from which ancestor the gene was inherited at each reticulate event. From a computational perspective, the tree containment problem lies at the foundation of the reconstruction of phylogenetic networks. In its classic version, where only topologies are considered, the problem is NP-hard [27], but for some specific classes of networks it can be solved in polynomial time [25].

Intuitively, an advantage of considering branch lengths is that it should allow one to locate more precisely a gene history within a network, and, more generally, it should give more specific answers to the tree containment problem. For example, whereas a tree topology may be contained in multiple different locations inside a network [9], this will happen much more rarely when branch lengths are taken into account (see, e.g., T_1 in Fig. 1). Similarly, some genes may only be detected to be inconsistent with a network when the branch lengths of their phylogenetic trees are considered (see, e.g., T_2 in Fig. 1). In practice, some uncertainty in the branch length estimates is to be expected, which implies that deciding whether a tree is contained in a network will depend on the confidence in these estimates (e.g., T_2 in Fig. 1 is only displayed by N if we allow its branch lengths to deviate by 2 or more units from their specified values).

While the possibility of having more meaningful answers to a computational problem is certainly an important advantage, another factor to consider is the complexity of calculating its solutions. It is known that adding constraints on branch lengths can lead to polynomial tractability of other problems in phylogenetics that would otherwise be NP-complete [13]. In this paper,

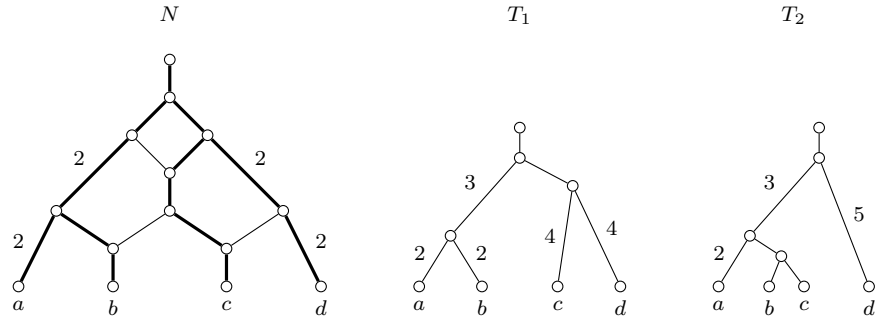


Fig. 1 Toy example on the impact of branch lengths on locating a tree within a network. If lengths are not taken into account, both T_1 and T_2 are displayed by N. Moreover, locating uniquely T_1 within N is not possible: there are 4 *switchings* (formally defined in the Preliminaries) of N for T_1, and 3 different ways to locate (*images* of) T_1 within N. If instead lengths are taken into account, the only image of T_1 within N is the one highlighted in bold, and T_2 is not displayed by N (in fact, the tree displayed by N isomorphic to T_2 has significantly different branch lengths). *Note*: branches with no label are assumed to have length 1.

we will show a number of results on the effect of taking into account branch lengths on the computational complexity of the tree containment problem. We first introduce the necessary mathematical preliminaries (Sec. 2), including a formal definition of the main problem that we consider (Tree Containment with Branch Lengths – TCBL), and of some variations of this problem accounting for the fact that branch lengths are usually only imprecise estimates of their true values (relaxed-TCBL and closest-TCBL). We then show a number of hardness (negative) results for the most general versions of these problems (Sec. 3), followed by a number of positive results (Sec. 4). Specifically, a suite of polynomial-time, pseudo-polynomial time and fixed parameter tractable algorithms that solve the problems above for networks of limited complexity (measured by their level [8,26]; definition below) and containing no unnecessary complexity (no redundant blobs [24]; also defined below).

2 Preliminaries

We define a *phylogenetic network* on X as a rooted directed acyclic graph with exactly one vertex of indegree 0 (the *root*), with no vertices with indegree and outdegree 1, and whose outdegree 0 vertices (the *leaves*) are bijectively labeled by the elements of X (the *taxa*). A *phylogenetic tree* is a phylogenetic network whose underlying undirected graph has no cycles. We consider phylogenetic networks (and thus trees) where each arc has an associated length. Formally, given an arc (u, v) of a phylogenetic network N, its *length* $\lambda_N(u, v)$ is a positive integer, i.e. strictly greater than zero. In this paper we will use the terms "arc lengths" and "branch lengths" interchangeably.

A phylogenetic tree or network is *binary* if all vertices have indegree 1 and outdegree 2 (*bifurcations*), indegree 2 and outdegree 1 (*reticulations*), indegree 0 and outdegree 1 (root) or indegree 1 and outdegree 0 (leaves). For example, all networks and trees in Fig. 1 are binary.

A *biconnected component* is a maximal connected subgraph that remains connected after removal of any one vertex. A *blob* of a phylogenetic network N is a biconnected component in which the undirected graph underpinning the biconnected component contains at least one cycle. Note that if a biconnected component of N is not a blob, then it is simply a cut arc (i.e., an arc whose removal disconnects N). The *level* of a binary phylogenetic network N is the maximum number of reticulations in any blob of N. An *outgoing arc* of a blob B is an arc (u, v) such that u is in B but v is not. An *incoming arc* (u, v) of B is such that v is in B but u is not. Note that a blob has at most one incoming arc. A blob is *redundant* if it has fewer than two outgoing arcs (i.e., one outgoing arc if the network is binary). As an example of these notions, the network N in Fig. 1 contains only one blob, which has 4 outgoing arcs and is thus non-redundant. Because this blob has 3 reticulations, N is level-3.

We say that two phylogenetic trees T_1 and T_2 are *isomorphic*, or that they have the same *topology*, if there is a bijection between the nodes that is both edge-preserving and leaf label-preserving. (Note that arc lengths do not play a role here.)

Given a phylogenetic tree T and a phylogenetic network N whose leaves are labeled bijectively by the same set X, we say that T *is displayed by* N *taking into account lengths*, if T can be obtained from N in the following way:

- for each reticulation, remove all incoming arcs except one; the tree obtained after this process is called a *switching* of N;
- repeat as long as possible the following *dummy leaf deletions*: for each leaf not labeled by an element of X, delete it;
- repeat as long as possible the following *vertex smoothings*: for each vertex v with exactly one parent p and one child c, replace it with an arc from p to c, with $\lambda_N(p, c) = \lambda_N(p, v) + \lambda_N(v, c)$.

In the following, we sometimes only say that T *is displayed by* N (with no mention of lengths) to mean that arc lengths are disregarded, and only topological information is taken into account.

Note that N displays T taking into account lengths if and only if there exists a subtree T' of N with the same root as N such that T can be obtained by repeatedly applying vertex smoothings to T'. In this case T' is said to be the *image* of T. There is a natural injection from the vertices of T to the vertices of T', so the definition of image extends naturally to any subgraph of T. In particular, the image of any arc in T is a path in N. Note that T can potentially have many images in N, but for a switching S of N, the image of T within S, if it exists, is unique. As an example of these notions, consider again Fig. 1, where N displays both T_1 and T_2, but only T_1 if lengths are taken into account. The part of N in bold is both a switching and an image of T_1 (as no dummy leaf deletions are necessary in this case).

Finally, is worth noting that, in this paper, if N displays T taking into account lengths, then the image of the root of T will always coincide with the root of N (no removal of vertices with indegree 0 and outdegree 1 is applied to obtain T). The biological justification for this is that trees and networks are normally rooted using an outgroup, which is sometimes omitted from the phylogeny; if arc lengths are taken into account, then the length of the path to the root of N in a tree displayed by N conveys the information regarding the distance from the outgroup. (See also [33] for a full discussion about this point.)

In this paper, we consider the following problem:

Problem 1 Tree Containment with Branch Lengths (TCBL)

Input: A phylogenetic network N and a phylogenetic tree T on the same set X, and both with positive integer arc lengths.

Output: YES if T is displayed by N taking into account lengths, NO otherwise.

We also consider two variations of TCBL seeking trees displayed by N that are allowed to somehow deviate from the query tree, to account for uncertainty in the branch lengths of the input tree. The first of these two problems aims to determine the existence of a tree displayed by N, whose branch lengths fall within a specified (confidence) interval.

Problem 2 relaxed-TCBL

Input: A phylogenetic network N with positive integer arc lengths, and a phylogenetic tree T, whose arcs are labelled by two positive integers $m_T(a)$ and $M_T(a)$, representing respectively the minimum and the maximum arc length. Both N and T are on the same set X.

Output: YES if and only if there exists a tree $\widetilde{T}$ displayed by N, isomorphic to T, and such that, for each arc a of T:

$$\lambda_{\widetilde{T}}(\tilde{a}) \in [m_T(a), M_T(a)]\,,$$

where $\tilde{a}$ denotes the arc in $\widetilde{T}$ that corresponds to a in T.

The second variation of TCBL we consider here, seeks – among all trees displayed by the network, and that are isomorphic to the input tree T – one that is closest to T, in terms of the maximum difference between branch lengths. There are several other alternative choices for defining the "closest" tree to T, for example if distance is measured in terms of the average difference between branch lengths. Later on, we will see that our results on this problem also apply to many of these alternative formulations (see Theorem 7).

Problem 3 CLOSEST-TCBL

Input: A phylogenetic network N and a phylogenetic tree T on the same set X, and both with positive integer arc lengths.
Output: A tree $\widetilde{T}$ displayed by N, isomorphic to T, that minimizes

$$\max \left| \lambda_T(a) - \lambda_{\widetilde{T}}(\tilde{a}) \right|,$$

where the max is over any choice of an arc a in T, and $\tilde{a}$ denotes the arc in $\widetilde{T}$ that corresponds to a in T. If no tree isomorphic to T is displayed by N, then report FAIL.

Note that all problems in this paper involving positive integer arc lengths are equivalent to problems where arc lengths are positive rational numbers: it suffices to multiply those rational numbers by the least common denominator of the fractions corresponding to these numbers in order to obtain integers.

We conclude with some definitions concerning computational complexity. An NP-complete decision problem that includes numbers in the input may or may not permit a *pseudo-polynomial time* algorithm. This is an algorithm which runs in polynomial time if the numbers in the input are encoded in unary, rather than binary. Formally speaking such algorithms are not polynomial time, since unary encodings artificially inflate the size of the input. Nevertheless, a pseudo-polynomial time algorithm has the potential to run quickly if the numbers in the input are not too large. An NP-complete problem with numbers in the input is said to be *strongly* NP-complete if it remains NP-complete even under unary encodings of the numbers. Informally, such problems remain intractable even if the numbers in the input are small. An NP-complete problem is *weakly* NP-complete if it is NP-complete when the numbers are encoded in binary. Summarizing, if one shows that a weakly NP-complete problem also permits a pseudo-polynomial time algorithm, then (under standard complexity assumptions) this excludes strong NP-completeness. Similarly, demonstrating strong NP-completeness excludes (under standard complexity assumptions) the existence of a pseudo-polynomial time algorithm. We refer to Garey and Johnson [16] for formal definitions.

On a slightly different note, an algorithm is said to be *fixed parameter tractable* (FPT) if it runs in time $O(f(k) \cdot poly(n))$ where n is the size of the input, k is some parameter of the input (in this article: the level of the network) and f is some computable function that depends only on k. An FPT algorithm for an NP-complete problem has the potential to run quickly even when n is large, as long as the parameter k is small, for example when f is a function of the form c^k, where c is a small constant greater than 1. We refer to [12,17] for more background on FPT algorithms.

3 Negative results

3.1 Strong NP-completeness

Theorem 1 *TCBL is strongly NP-complete, even when the phylogenetic tree T and the phylogenetic network N are binary.*

Proof We reduce to TCBL the following 3-PARTITION problem, which is strongly NP-complete [15]:

Input: an integer Σ and a multiset S of $3m$ positive integers n_i in $]\Sigma/4, \Sigma/2[$ such that $m\Sigma = \sum_{i\in[1..3m]} n_i$.

Output: YES if S can be partitioned into m subsets of elements $S_1, S_2, \ldots, S_m$ each of size 3, such that the sums of the numbers in each subset are all equal; NO otherwise.

Let us consider a multiset S containing $3m$ positive integers n_i which have sum $m\Sigma$.

We build a phylogenetic tree T in the following way. We first build a directed path containing $m+2$ vertices, whose arcs all have length 1. We call its initial vertex ρ, its final vertex b_0, and the ancestors of b_0, from the parent of b_0 to the child of ρ are called v_1 to v_m. Then, to each of the m vertices v_i for $i \in [1..m]$ on this directed path, from bottom to top, we add an arc of length $L = \Sigma + 6m^2 - 3m + 1$ to a child, called b_i.

We now build a phylogenetic network N in the following way. We start by creating a copy of T but for each $i \in [1..m]$ we remove the arc (v_i, b_i) and replace it by an arc of length 1 from v_i to a new vertex r_1^i (see Figure 2). Then we create $3m$ subnetworks called B_k, for $k \in [1..3m]$, as described in Figure 3. For ease of notation, we consider that vertex p_k^2 is also labeled p_k^1 and c_k^2 is also labeled c_k^1 for any $k \in [1..3m]$. Finally, we add arcs (b_k^i, r_{k+1}^i) of length 1 for each $k \in [1..3m-1]$ and $i \in [1..m]$ (to connect each B_k with B_{k+1}) and arcs of length 1 from b_{3m}^i to b_i for each $i \in [1..m]$ to obtain N.

Suppose that S can be partitioned into m subsets of elements $S_1, S_2, \ldots, S_m$ each of size 3, such that the sums of the numbers in each subset are all equal to Σ. We now prove that this implies that T and N constructed above constitute a positive instance of TCBL.

For each n_k, if it belongs to S_i then we remove from N all arcs (c_k^j, b_k^j) for $j \in [1..m] - \{i\}$, as well as all arcs (r_k^j, p_k^j) for $j \in [1..m] - \{i\} - \{1 \text{ if } i \neq 2\}$, the arc (r_k^i, b_k^i), and finally the arc (p_k^{i-1}, p_k^i) if $i \notin \{1, 2\}$. This way, we obtain a switching T' of N for T, shown in Figure 3(b).

In T', the only path from r_k^i to b_k^i goes through the arc (p_k^m, c_k^m) of length n_k, so the total length of this path is $2m - 2 + n_k$. For all other S_j, $j \in [1..k] - \{i\}$, the only directed path from r_k^i to b_k^i is an arc of length $2m - 2$. Thanks to the arcs (b_k^j, r_{k+1}^j), for $j \in [1, m]$ a unique path can be found in T' from v_j to b_j. We can check that the lengths of the arcs of T leading to b_i with $i \in [1..m]$ are consistent with the lengths of these paths: the latter have all

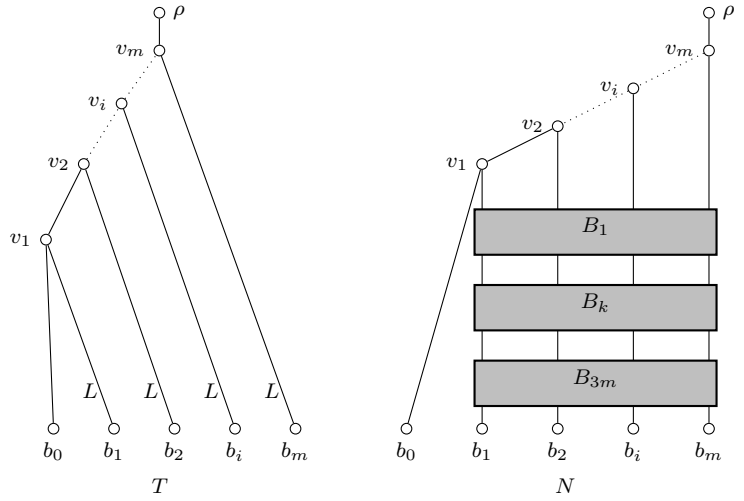


Fig. 2 The tree T and the network N used in the proof of Theorem 1. All arcs are directed downwards. The dotted arcs represent parts of the network which are not shown in details but which ensure connectivity. All arcs incident to leaves b_i of T, for $i \in [1..m]$ have length $L = \Sigma + 6m^2 - 3m + 1$; and remaining arcs of T have length 1. All arcs of N have length 1, except in the $3m$ boxes B_k (see Figure 3(a) for more details on the content of those $3m$ boxes B_k).

length $3m((2m-2)+1)+(\sum_{n_k \in S_i} n_k)+1 = \Sigma+6m^2-3m+1$. Furthermore, all other arcs of T (on the path from ρ to b_0) are also present in T' with the same configuration and length, meaning that, as we wished to prove, T is displayed by N taking into account lengths.

We now focus on the converse, supposing that the tree T is displayed by N taking into account lengths. We first note that any switching T' of N for T contains the vertices b_0, v_i for $i \in [1..m]$, ρ and the arcs between these vertices. Furthermore, T' also contains a path $P_i(T')$ from v_i to b_i, for each $i \in [1..m]$, of length L.

Claim 1: For any switching T' of N for T, for any $i \in [1..m]$ and $k \in [1..3m]$, $r_k^i \in P_i(T')$ and $b_k^i \in P_i(T')$.

We prove it by induction on k. For $k = 1$, for all $i \in [1..m]$, vertex r_1^i has indegree 1 and its unique parent is contained in $P_i(T')$ so it is also contained in $P_i(T')$. As arc (p_1^m, c_1^m) belongs to all paths between p_1^i and c_1^j for $i, j \in [1..m]$, at most one of the paths $P_i(T')$ contains (p_1^m, c_1^m). If no such path exists then all paths $P_i(T')$ contain arc (r_1^i, b_1^i), so $b_1^i \in P_i(T')$. Otherwise, we denote by $P_{i_0}(T')$ the path containing (p_1^m, c_1^m). All other paths $P_i(T')$ for $i \in [1..m] - i_0$ contain arc (r_1^i, b_1^i), so $b_1^i \in P_i(T')$. Because none of those paths contain $b_1^{i_0}$, we must have $b_1^{i_0} \in P_{i_0}(T')$. Therefore, for all $i \in [1..m]$, $b_1^i \in P_i(T')$.

Supposing vertices r_{k-1}^i and b_{k-1}^i belong to $P_i(T')$ for all $i \in [1..m]$, we can reproduce the proof above by replacing "1" by "k" each time we refer to

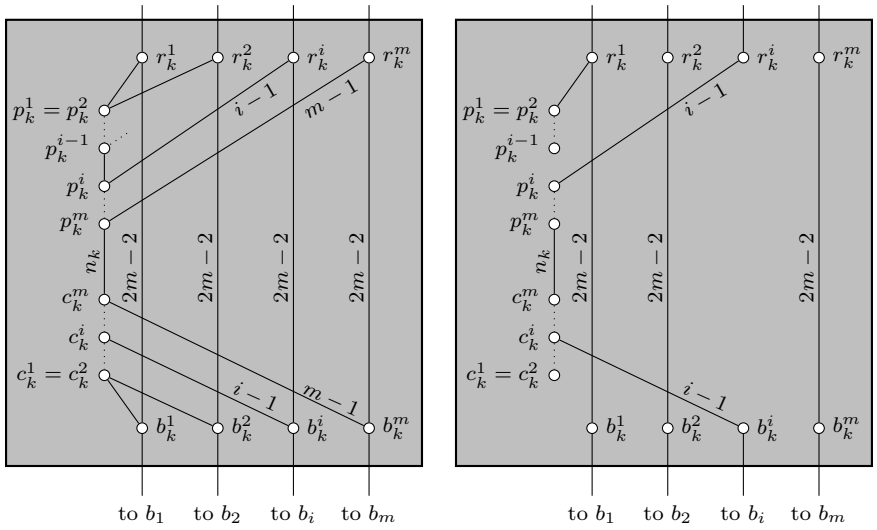


Fig. 3 The content of the box B_k (a) and a corresponding switching (b) of the network of Fig. 2. All arcs are directed downwards. The dotted arcs represent parts of the network which are not shown in details but which ensure connectivity. All arcs have length 1 except arcs (r_k^i, b_k^i) for $i \in [1..m]$ which have length $2m-2$, arcs (r_k^i, p_k^i) and (c_k^i, b_k^i), for $i > 1$, which have length $i-1$, and the arc (p_k^m, c_k^m) with length n_k.

b_1^i, c_1^i, p_1^i and r_1^i for any $i \in [1..m]$, in order to deduce that r_k^i and b_k^i belong to $P_i(T')$.

Claim 2: For any switching T' of N for T, for any $k \in [1..3m]$, one of the paths $P_i(T')$ contains arc (p_k^m, c_k^m).

First, using Claim 1, we can consider each portion of the path $P_i(T')$ from r_k^i to b_k^i in T', and note that this portion has length $2m-2+n_k$ if $P_i(T')$ contains arc (p_k^m, c_k^m), or length $2m-2$ otherwise.

Therefore, supposing by contradiction that there exists at least one $k_0 \in [1..3m]$ such that none of the paths $P_i(T')$ contain arc $(p_{k_0}^m, c_{k_0}^m)$, then the cumulative length L_{k_0} of the portions of all paths $P_i(T')$ between $r_{k_0}^i$ and $b_{k_0}^i$, for $i \in [1..m]$, is $m(2m-2)$. Therefore, summing the lengths of all these portions and the ones of arcs (b_k^i, r_{k+1}^i) between them as well as the ones of the arcs (v_i, r_1^i) and (b_{3m}^i, b_i) for any $i \in [1..m]$, the sum L' of the lengths of all paths $P_i(T')$ for $i \in [1..m]$ is at most $m+3m(L_{k_0}+m)+(\sum_{k\in[1..3m]} n_k)-n_{k_0} = m(6m^2-3m+1+\Sigma)-n_{k_0} = mL-n_{k_0}$. However, the sum L_T of the lengths of all arcs (v_i, b_i) of T is equal to mL so $L' < L_T$, meaning that T is not displayed by N taking into account lengths: contradiction.

Claim 3: for any switching T' of N for T, for any $i \in [1..m]$, there are exactly 3 arcs of the form (p_k^m, c_k^m) contained in $P_i(T')$.

We suppose by contradiction that there exists $i \in [1..m]$, and k_1, k_2, k_3 and $k_4 \in [1..3m]$ such that $(p_{k_1}^m, c_{k_1}^m)$, $(p_{k_2}^m, c_{k_2}^m)$, $(p_{k_3}^m, c_{k_3}^m)$ and $(p_{k_4}^m, c_{k_4}^m)$ are

contained in $P_i(T')$. Then, this path has length at least $n_{k_1} + n_{k_2} + n_{k_3} + n_{k_4} + 3m(2m-2) + 3m + 1 > \Sigma + 3m(2m-1) + 1$ because $n_i > \Sigma/4$ for all $i \in [1..3m]$. So T' contains a path from v_i to b_i which is strictly longer than the arc from v_i to b_i in T, so T is not displayed by T', nor in N: contradiction.

Now, we suppose by contradiction that there exists $i \in [1..m]$ such that P_i contains at most 2 arcs of the form (p_k^m, c_k^m). Then, according to Claim 2, each of the the remaining $3m-2$ arcs of the form (p_k^m, c_k^m) must be contained by one of the remaining $m-1$ paths P_j for $j \in [1..m] - \{i\}$. So at least one of those paths must contain strictly more than 3 such arcs, which contradicts the previous paragraph: contradiction.

Finally, for any switching T' of N for T, the fact that T is displayed by N taking into account lengths, implies that the length of each arc (v_i, b_i) of T, $\Sigma + 6m^2 - 3m + 1$, equals the length of each path $P_i(T')$. Claim 2 and 3 imply that the arcs of the form (p_k^m, c_k^m) are partitioned into the paths $P_i(T')$, with each $P_i(T')$ containing exactly 3 such arcs. Denoting by n_{k_i}, $n_{k'_i}$ and $n_{k''_i}$ the length of such arcs, we obtain that the length of $P_i(T')$ equals $n_{k_i} + n_{k'_i} + n_{k''_i} + 6m^2 - 3m + 1$, therefore $n_{k_i} + n_{k'_i} + n_{k''_i} = \Sigma$, which implies that S can be partitioned into m subsets of elements $S_i = \{n_{k_i}, n_{k'_i}, n_{k''_i}\}$, such that the sums of the numbers in each subset S_i are all equal to Σ.

Finally, it is easy to see that the problem is in NP: a switching T' of the input network N is a polynomial size certificate of the fact that the input tree T is contained in N. We can check in polynomial time that T can be obtained from T' by applying dummy leaf deletions and vertex smoothings until possible, and checking that the obtained tree is isomorphic with T. □

We note that Theorem 1 can be extended to binary tree-sibling [6] time-consistent [3] networks, by multiplying by 2 all arc lengths of the network constructed in the proof (in order to keep integer arc lengths even if those arcs are subdivided, which happens at most once), and using a gadget shown in Figure 4, adapted from Fig. 4 of [25] with arcs of length 1, and the operations described in the proof of Theorem 3 of the same article.

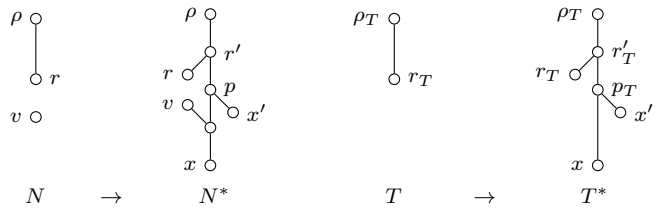


Fig. 4 How our slightly modified HangLeaves(v) modifies N and T. Vertices ρ and ρ_T are the roots of N and T respectively. All arcs have length 1, except (r', r) of N^* which has the same length as (ρ, r) of N, (r'_T, r_T) of T^* which has the same length as (ρ_T, r_T) of T and (p_T, x) which has length 2.

Corollary 1 RELAXED-TCBL *is strongly NP-complete, and* CLOSEST-TCBL *is strongly NP-hard.*

Proof TCBL can be easily reduced to both problems. Indeed, any instance of TCBL corresponds to an instance of RELAXED-TCBL with $m_T(a) = M_T(a) := \lambda_T(a)$ for each arc of T. Additionally, TCBL can be reduced to CLOSEST-TCBL by checking whether there exists a solution $\widetilde{T}$ with $\max|\lambda_T(a) - \lambda_{\widetilde{T}}(\tilde{a})| = 0$. □

3.2 Weak NP-completeness for level-2 networks

The strong NP-completeness result above does not imply anything about the hardness of TCBL on networks of bounded level. Unfortunately, TCBL is hard even for low-level networks, as we now show.

Theorem 2 *TCBL is weakly NP-complete for level-2 binary networks.*

Proof First, recall that TCBL is in NP (Theorem 1). To prove the theorem, we will reduce from the SUBSET SUM problem: given a multiset of positive integers $I = \{n_1, \ldots, n_k\}$ and a positive integer s, is there a non-empty subset of I whose sum is s? The SUBSET SUM problem is known to be weakly NP-complete.

Now, we show how to construct an instance of the TCBL problem with the required characteristics, for each instance of the SUBSET SUM problem. This can be done by defining the tree T and the network N as follows. The tree T is defined as the rooted tree on two leaves labeled a and b, parent ρ' and root ρ, and arcs (ρ, ρ'), (ρ', a) and (ρ', b), respectively of length 1, 1 and $s + 3k + 1$. The network N is the network on the two leaves labeled a and b shown in Fig. 5, where $L > s + 3k + 1$. Then, it is easy to see that a positive instance of the TCBL problem gives a positive instance of the SUBSET SUM problem through the previous transformation, and vice versa. This is true because no switching S of N giving rise to T will ever contain the arcs with length L. Thus, the paths in S going through the blob containing the arc with length n_i can have either length 2 or $2 + n_i$. Now, any path from ρ'_N to the leaf labeled b has to go through all blobs, and through all arcs connecting these blobs. The sum of the lengths of the arcs on this path but outside the blobs is $k + 1$. Thus, there exists a path from ρ'_N to b with length $s + 3k + 1$ if and only if there is a non-empty subset of $I = \{n_1, \ldots, n_k\}$ whose sum is s.

As to the weakness of this NP-completeness result, we refer to Section 4.2, where we give a pseudo-polynomial algorithm for TCBL on any binary network of bounded level. □

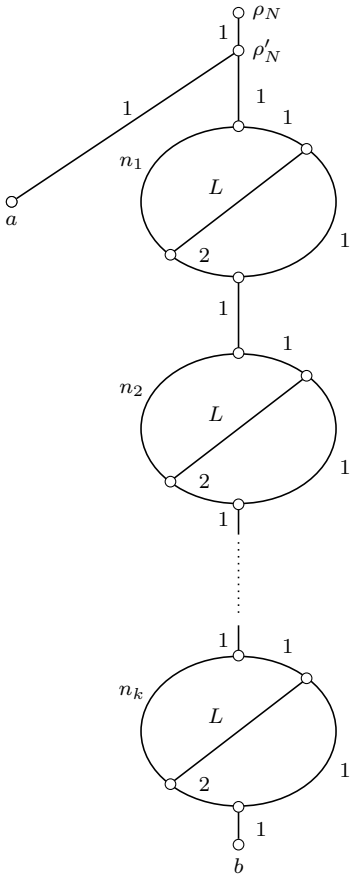


Fig. 5 The network used in the proof of Theorem 2.

4 Positive results

4.1 TCBL is FPT in the level of the network when no blob is redundant

Note that in the weak NP-completeness result from Section 3.2 the blobs have only one outgoing arc each – that is, they are redundant. If we require that every blob has at least two outgoing arcs, then dynamic programming becomes possible, and the problem becomes much easier. The high-level reason for this as follows. Because blobs in the network N have at least two outgoing arcs, the image of any tree displayed by N will branch at least once inside each blob. This means that for each arc (u', v') of T, if the image of u' lies inside a blob B, then the image of v' either lies (i) also inside B or (ii) in one of the biconnected components C_i immediately underneath B. This last observation holds with or without arc lengths, but when taking lengths into account it has an extra significance. Indeed, suppose N displays T taking lengths into

account, and S is a switching of N that induces the image of T inside N. Let (u', v') be an arc of T. If, within S, the image of u' lies inside a blob B and the image of v' lies inside a biconnected component C_i immediately underneath B, then the image of the arc (u', v') – a path – is naturally partitioned into 3 parts. Namely, a subpath inside B (starting at the image of u'), followed by an outgoing arc of B, followed by a subpath inside C_i (terminating at the image of v'). See Fig. 6 for an illustration. Within S, the lengths of these 3 parts must sum to $\lambda_T(u', v')$. The dynamic programming algorithm described below, in which we process the blobs in a bottom-up fashion, makes heavy use of this insight.

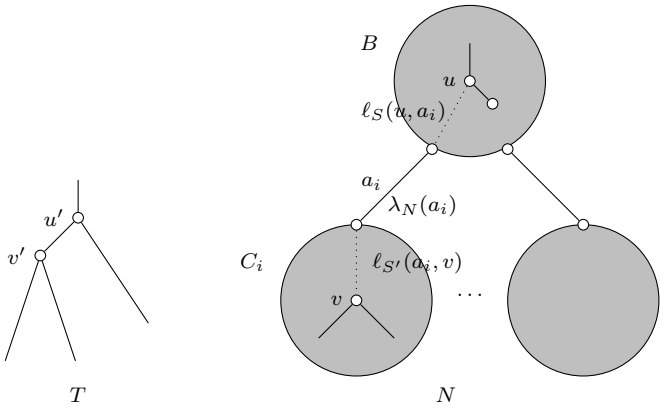


Fig. 6 Illustration of the idea at the basis of Algorithm 1. If a network N displays a tree T and the image u of u' (for an arc (u', v') of T) lies inside a blob B of N, then – assuming every blob of the network has at least two outgoing arcs – the image v of v' will either lie inside B, or inside a blob C_i that is immediately beneath B. In the latter case the image of (u', v') can be naturally partitioned into three parts, as shown. This is the foundation for the dynamic programming approach used in Theorem 3 and later in Theorems 5 and 6.

Theorem 3 *Let N be a level-k binary network and T be a rooted binary tree, both on X. If no blob of N is redundant, then TCBL can be solved in time $O(k \cdot 2^k \cdot n)$ using $O(k \cdot 2^k \cdot n)$ space, where $n = |X|$.*

Proof Firstly, note that networks can have nodes that are not inside blobs (i.e. tree-like regions). To unify the analysis, it is helpful to also regard such a node u (including when u is a taxon) as a blob with 0 reticulations: the definition of incoming and outgoing arcs extends without difficulty. Specifically, in this case they will simply be the arcs incoming to and outgoing from u. We regard such blobs as having exactly one switching.

Next, it is easy to see that the blobs of N can themselves be organized as a rooted tree, known as the *blobbed-tree* [14, 18]. In other words, the parent-child relation between blobs is well-defined, and unique. The idea is to process the

blobs in bottom-up, post-order fashion. Hence, if a blob B has blob children $C_1, C_2, \ldots$ (underneath outgoing arcs $a_1, a_2 \ldots$) we first process $C_1, C_2, \ldots$ and then B. Our goal is to identify some switching of B which can legitimately be merged with one switching each from $C_1, C_2, \ldots$. We initialize the dynamic programming by, for each blob B that is a taxon, recording that it has exactly one switching whose root-path has length 0. (The definition and meaning of root-path will be given in due course).

For each blob B that is not a taxon, we will loop through the (at most) 2^k ways to switch the reticulations within B. Some of these candidate switchings can be immediately discarded on topological grounds, i.e., such a switching of B induces bifurcations that are not present in T. Some other candidate switchings S can be discarded on the basis of the lengths of their internal paths, that is, the paths $u \to v$ entirely contained within S coinciding with the image of some arc (u', v') in T. Clearly the path $u \to v$ must have the same length as (u', v').

Finally, we need to check whether the candidate switching S can be combined with switchings from $C_1, C_2, \ldots$ such that arc lengths are correctly taken into account. This proceeds as follows. Observe that, for each outgoing arc a_i of B, a_i lies on the image of an arc (u', v') of T. This arc of T is uniquely defined. Let u be the image of u' in B, and let $\ell_S(u, a_i)$ be the total length of the path (in S) from u to the tail of a_i. The image of v' will lie somewhere inside C_i. For a switching S' of C_i, let v be the image of v' within S', and let $\ell_{S'}(a_i, v)$ be the total length of the path (in S') from the head of a_i to v. (See Fig. 6).

If we wish to combine S' with S, then we have to require $\lambda_T(u', v') = \ell_S(u, a_i) + \lambda_N(a_i) + \ell_{S'}(a_i, v)$. To know whether such an S' exists, B can ask C_i the question: "do you have a candidate switching S' such that $\ell_{S'}(a_i, v) = \lambda_T(u', v') - \ell_S(u, a_i) - \lambda_N(a_i)$?" This will be true if and only if C_i has a candidate switching S' such that the *root-path* in S' – defined as the path from the root of C_i to the first branching node of S' – has length exactly $\lambda_T(u', v') - \ell_S(u, a_i) - \lambda_N(a_i)$. (We consider a node of S' to be a branching node if it is the image of some node of T.) B queries all its children $C_1, C_2, \ldots$ in this way. If all the C_i answer affirmatively, then we store S, together with the length of its root-path, as a candidate switching of B, otherwise we discard S.

This process is repeated until we have finished processing the highest blob B of N. The answer to TCBL is YES, if and only if this highest blob B has stored at least one candidate switching. Pseudocode formalizing the description above is provided in Algorithm 1.

We now analyse the running time and storage requirements. For step 1, observe that the blobbed-tree can easily be constructed once all the biconnected components of the undirected, underlying graph of N have been identified. The biconnected components can be found in linear time (in the size of the graph) using the well-known algorithm of Hopcroft and Tarjan (see, e.g., [10]). Because every blob has at least two outgoing arcs, N will have $O(kn)$ vertices and arcs, (see, e.g., Lemma 4.5 in [23] and discussion thereafter) so

the time to construct the blobbed-tree is at most $O(kn)$. Moreover, N has $O(n)$ blobs, meaning that the blobbed-tree has $O(n)$ nodes and that step 2 can be completed in $O(n)$ time by checking whether T and the blobbed-tree are compatible. (The compatibility of two trees can be tested in linear time [37].) Each blob has at most 2^k switchings, and each switch can be encoded in k bits. If we simply keep all the switchings in memory (which can be useful for constructing an actual switching of N, whenever the answer to TCBL is YES) then at most $O(k \cdot 2^k \cdot n)$ space is required.

For time complexity, each blob B loops through at most 2^k switchings, and for each switching S it is necessary to check the topological legitimacy of S (step 3(a)), that internal paths of the switching have the correct lengths (step 3(b)), and subsequently to make exactly one query to each of its child blobs C_i (step 3(c)). We shall return to steps 3(a) and 3(b) in due course. It is helpful to count queries from the perspective of the blob that is queried. In the entire course of the algorithm, a blob will be queried at most 2^k times. Recalling that the number of blobs is $O(n)$, in total at most $O(2^k \cdot n)$ queries will be made, so the total time devoted to queries is $O(q \cdot 2^k \cdot n)$, where q is the time to answer each query. Recall that a query consists of checking whether a blob has a switching whose root-path has a given length. Each blob needs to store at most 2^k switchings. By storing these switchings (ranked by the lengths of their root-paths) in a balanced look-up structure (e.g. red-black trees) an incoming query can be answered in logarithmic time in the number of stored switchings, that is, in time $\log 2^k = k$. Hence, the total time spent on queries is $O(k \cdot 2^k \cdot n)$.

For steps 3(a) and 3(b) we require amortized analysis. Let $d^+(B)$ denote the number of outgoing arcs from blob B. The blob B can be viewed in isolation as a rooted phylogenetic network with $d^+(B)$ "taxa", so inside B there are $O(k \cdot d^+(B))$ vertices and arcs [23]. Therefore, the time to convert a switching S from B into a tree T' on $d^+(B)$ "taxa" (via dummy leaf deletions and vertex smoothings) is at most $O(k \cdot d^+(B))$. The topology and internal arc lengths of T' can be checked against those of the corresponding part of T in $O(d^+(B))$ time [37]. Hence, the total time spent on steps 3(a) and 3(b) is

$$\sum_B O(2^k \cdot k \cdot d^+(B)), \tag{1}$$

where the sum ranges over all blobs. Note that $\sum_B d^+(B)$ is $O(n)$ because there are $O(n)$ blobs and each outgoing arc enters exactly one blob. Hence, the expression (1) is $O(k \cdot 2^k \cdot n)$, matching the time bound for the queries. Hence, the overall running time of the algorithm is $O(k \cdot 2^k \cdot n)$. □

Algorithm 1 FPT algorithm for TCBL on networks with no redundant blobs

1. Decompose N into blobs and construct the blobbed-tree T_N, whose nodes are the blobs in N and whose arcs are the arcs external to the blobs.
2. Check that T_N is compatible with the input tree T (in fact check that T_N can be obtained from T via arc contractions). If this is not the case, then terminate with a NO. Otherwise each vertex B in T_N can be obtained as the contraction of a subtree $T(B)$ of T, and each arc a in the blobbed-tree T_N originates from an arc a' in T. Store references to the a' and the $T(B)$.
3. **for each** blob B, in bottom-up order:
   for each switching S of B:
   (a) check that S is topologically compatible with $T(B)$.
   (b) check that each arc of $T(B)$ is as long as its image in S;
   (c) **for each** blob C_i that is a child of B, via the arc a_i:
   – check that C_i has stored a switching S' whose root-path has the appropriate length. Specifically, we require $\ell_{S'}(a_i, v) = \lambda_T(u', v') - \ell_S(u, a_i) - \lambda_N(a_i)$, where (u', v') is the arc of T on whose image a_i lies (i.e. a_i'), and u and v are the uniquely defined images of u' and v' in S and S', respectively.
   (d) if none of the checks above failed, store S along with the length of its root-path;
   (e) if no switching is stored for B, then terminate with NO, as no tree displayed by N satisfies the requirements.
4. If the algorithm gets this far, then it returns YES and the image in N of T can be obtained by combining a switching S stored for the root blob, to the switchings S' found for its child blobs, recursively.

4.2 Pseudo-polynomial solution of TCBL on any network of bounded level

Redundant blobs are problematic for TCBL because when they appear "in series" (as in Fig. 5) they give rise to an exponential explosion of paths that can be the images of an arc a in T, and, as we saw, checking the existence of a path of the appropriate length $\lambda_T(a)$ is at least as hard as SUBSET SUM. Just like for SUBSET SUM, however, a pseudo-polynomial time solution is possible, as we now show.

Theorem 4 *Let N be a level-k binary network with b blobs, and let T be a rooted binary tree on the same set of n taxa. TCBL can be solved in time $O(k \cdot b \cdot L + 2^k \cdot n \cdot L)$ using $O(k \cdot n \cdot L)$ space, where L is an upper bound on arc lengths in T.*

Proof The algorithm we now describe is based on the following two observations (we use here the same notational conventions as in Algorithm 1). First,

if T is indeed displayed by N, the image $u \to v$ of any of its arcs (u', v') will either be entirely contained in one blob, or u and v will be in two different blobs, which can only be separated by redundant blobs. Second, it only makes sense to store a switching S' of a blob C_i, if $\ell_{S'}(a_i, v) < \lambda_T(u', v')$, i.e., if its root-leaf path is shorter than the corresponding arc in T, meaning that we only need to store $O(L)$ switchings per blob.

Accordingly, we modify Algorithm 1 as follows:

Step 2a: Check that T_N is compatible with the input tree T in the following way: replace any chain of redundant blobs $M_1, M_2, \ldots, M_h$ in T_N with a single arc from the parent of M_1 to the child of M_h, and then check that the resulting blobbed-tree T'_N can be obtained from T via arc contractions. If this is not the case then terminate with a NO. Otherwise for each arc a and vertex B in T'_N, define and store a' and $T(B)$ as before.

Step 2b: For each arc a in T'_N, calculate a set of lengths $L(a)$ as follows. If a originates from a chain of redundant blobs $M_1, M_2, \ldots, M_h$, then $L(a)$ is obtained by calculating the lengths of all paths in N starting with the incoming arc of M_1 and ending with the (unique) outgoing arc of M_h. Only keep the lengths that are smaller than $\lambda_T(a')$. For the remaining arcs in T'_N, simply set $L(a) := \{\lambda_N(a)\}$.

The algorithm only visits non-redundant blobs, performing a bottom-up traversal of T'_N, and doing the same as Algorithm 1 except for:

Step 3c: **for each** blob C_i that is a child of B in T'_N :

- check the existence of an $\ell \in L(a_i)$ and a switching S' stored for C_i that satisfy:

$$\ell_{S'}(a_i, v) = \lambda_T(u', v') - \ell_S(u, a_i) - \ell. \tag{2}$$

To complete the description of the algorithm resulting from these changes, we assume that the switchings for a (non-redundant) blob B are stored in an array S_B indexed by the root-path length of the switching. If two or more switchings of a blob have the same root-path length ℓ, we only keep one of them in $S_B[\ell]$. Because for C_i we only store the switchings whose root-path length is less than $\lambda_T(a'_i)$, the S_B arrays have size $O(L)$.

As for step 2b above, the computation of $L(a)$ for an arc in T'_N corresponding to a chain of redundant blobs can be implemented in a number of ways. Here we assume that the vertices in $M_1, M_2, \ldots, M_h$ are visited following a topological ordering, and that, for each visited vertex v, we fill a boolean array P_v of length $\lambda_T(a')$, where $P_v[\ell]$ is true if and only if there exists a path of length ℓ from the tail of the arc incoming M_1 to v. Once P_{v_h} for the head v_h of the arc outgoing M_h has been filled, $L(a)$ will then be equal to the set of indices ℓ for which $P_{v_h}[\ell]$ is true.

We are now ready to analyse the complexity of this algorithm. We start with the space complexity. First note that every redundant blob of level k in a binary network must have exactly $2k$ vertices (as the number of bifurcations must equal the number of reticulations). Because each redundant blob has $O(k)$ vertices, and each P_v array is stored in $O(L)$ space, step 2b requires

$O(kL)$ space to process each redundant blob. Because every time a new redundant blob M_{i+1} is processed, the P_v arrays for the vertices in M_i can be deleted, step 2b only requires $O(kL)$ space in total. This however is dominated by the space required to store $O(L)$ switchings for each non-redundant blob. Since there are $O(n)$ non-redundant blobs in N and each switching requires $O(k)$ bits to be represented, the space complexity of the algorithm is $O(k{\cdot}n{\cdot}L)$.

We now analyse the time complexity. Checking the compatibility of the blobbed tree and T (step 2a) can be done in time $O(n+b)$, as this is the size of T'_N before replacing the chains of redundant blobs. The computation of the arrays P_v (step 2b) involves $O(L)$ operations per arc in $M_1, M_2, \ldots, M_h$. Because there are $O(b)$ redundant blobs, and because each of them contains $O(k)$ arcs, calculating all the P_v arrays requires time $O(k \cdot b \cdot L)$.

The other runtime-demanding operations are the queries in step 3c. These involve asking, for each $\ell \in L(a_i)$, whether C_i has a switching whose root-path has the length in Eqn. (2). Each of these queries can be answered in constant time by checking whether $S_{C_i}[\lambda_T(u',v') - \ell_S(u,a_i) - \ell\,]$ is filled or not. Because every non-redundant blob C_i will be queried at most $2^k \cdot L(a_i)$ times, and because there are $O(n)$ non-redundant blobs, the total time devoted to these queries is $O(2^k \cdot n \cdot L)$. The remaining steps require the same time complexities as in Theorem 3. By adding up all these runtimes we obtain a total time complexity of $O(k \cdot b \cdot L + 2^k \cdot n \cdot L)$. □

4.3 CLOSEST-TCBL and RELAXED-TCBL are FPT in the level of the network when no blob is redundant

We now show that Algorithm 1 can be adapted to solve the "noisy" variations of TCBL that we have introduced in the Preliminaries section.

Theorem 5 *Let N be a level-k binary network and T be a rooted binary tree, both on the same set of n taxa. The arcs of N are labelled by positive integer lengths, and the arcs of T are labelled by a minimum and a maximum positive integer length. If no blob of N is redundant, then* RELAXED-TCBL *can be solved in $O(k \cdot 2^k \cdot n)$ time and space.*

Proof We modify Algorithm 1 to allow some flexibility whenever a check on lengths is made: instead of testing for equality between arc lengths in the tree and the path lengths observed in the partial switching under consideration, we now check that the path length belongs to the input interval. Specifically, we modify two steps in Algorithm 1 as follows:

Step 3b: check that every arc (u',v') of $T(B)$, whose image is an internal path $u \to v$ of S, is such that $\ell_S(u,v) \in [m_T(u',v'), M_T(u',v')]$.

Step 3c: check that, among the switchings stored for C_i, there exists at least one switching S' whose root-path $\ell_{S'}(a_i,v)$ has a length in the appropriate interval. Specifically, using the same notation as in Algorithm 1, check that:

$$\ell_S(u,a_i) + \lambda_N(a_i) + \ell_{S'}(a_i,v) \quad \in \quad [m_T(a'_i), M_T(a'_i)],$$

that is:

$$m_T(a'_i) - \ell_S(u, a_i) - \lambda_N(a_i) \quad \leq \quad \ell_{S'}(a_i, v) \quad \leq \quad M_T(a'_i) - \ell_S(u, a_i) - \lambda_N(a_i). \tag{3}$$

We can use the same data structures used by Algorithm 1, so the space complexity remains $O(k \cdot 2^k \cdot n)$. As for time complexity, the only relevant difference is in step 3c: instead of querying about the existence of a switching with a definite path-length, we now query about the existence of a switching whose path-length falls within an interval (see Eqn. (3)). In a balanced look-up structure, this query can be answered again in time $O(\log 2^k) = O(k)$. In conclusion the time complexity remains the same as that in Theorem 3, that is $O(k \cdot 2^k \cdot n)$. □

Theorem 6 *Let N be a level-k binary network and T be a rooted binary tree, both with positive integer arc lengths and on the same set of n taxa. If no blob of N is redundant, then* CLOSEST-TCBL *can be solved in time $O(2^{2k} \cdot n)$ using $O(k \cdot 2^k \cdot n)$ space.*

Proof As we shall see, we modify Algorithm 1 by removing all checks on arc lengths, and by keeping references to those switchings that may become part of an optimal solution in the end: any topologically-viable switching S of a blob B is stored along with a reference, for each child blob C_i, to the switching S' that must be combined with S. Moreover, we compute recursively μ_S, which we define as follows:

$$\mu_S = \max \left| \lambda_T(a) - \lambda_{\widetilde{T}}(\tilde{a}) \right|,$$

where $\widetilde{T}$ is the subtree displayed by N obtained by (recursively) combining S to the switchings stored for its child blobs, and then applying dummy leaf deletions and vertex smoothings. The max is calculated over any arc $\tilde{a}$ in $\widetilde{T}$ and its corresponding arc a in T, excluding the root arc $\tilde{a}_r$ of $\widetilde{T}$ from this computation. This is because the length of the path above S, which must be combined with $\tilde{a}_r$, is unknown when S is defined.

In more detail, we modify Algorithm 1 as follows:

Step 3b: no check is made on the lengths of the internal paths of S; instead initialize μ_S as follows:

$$\mu_S := \max \left| \lambda_T(u', v') - \ell_S(u, v) \right|,$$

where the max is over all arcs (u', v') in $T(B)$, and u, v are the images of u', v' in S, respectively. Trivially, if B is just a vertex in N, the max above is over an empty set, meaning that μ_S can be initialized to any sufficiently small value (e.g., 0).

Step 3c: **for each** blob C_i that is a child of B:

- among the switchings stored for C_i, seek the switching S' minimizing

$$\max \{\mu_{S'}, |\ell_S(u, a_i) + \lambda_N(a_i) + \ell_{S'}(a_i, v) - \lambda_T(u', v')|\} \tag{4}$$

- set μ_S to $\max \{\mu_S, \text{value of (4) for } S'\}$

Step 3d: store S along with μ_S, with the length of its root-path, and with references to the child switchings S' minimizing (4)

Step 4: seek the switching S stored for the root blob that has minimum μ_S, and combine it recursively to the switchings S' found for its child blobs. In the end, a switching $\widetilde{S}$ for the entire network N is obtained, which can be used to construct $\widetilde{T}$.

The correctness of the algorithm presented above is based on the following observation, allowing our dynamic programming solution of the problem:

Observation. Let B be a blob of N, and C_i be one of its child blobs. If a switching S of B is part of an optimal solution to CLOSEST-TCBL, then we can assume that S must be combined with a switching S' of C_i that minimizes Eqn. (4). This means that even if there exists an optimal solution in which S is combined with S'', a non-minimal switching of C_i with respect to Eqn. (4), then we can replace S'' with S' and the solution we obtain will still be optimal.

Once again, space complexity is $O(k \cdot 2^k \cdot n)$, as the only additional objects to store are the references to the child switchings of S, and the value of μ_S for each the $O(2^k \cdot n)$ stored switchings. As for time complexity, each query within step 3c now involves scanning the entire set of $O(2^k)$ switchings stored for C_i, thus taking time $O(2^k)$ – whereas the previous algorithms only required $O(k)$ time. Since there are again $O(2^k \cdot n)$ queries to make, the running time is now $O(2^{2k} \cdot n)$. □

We conclude this section by noting that, if we reformulate CLOSEST-TCBL replacing the max with a sum, and taking any positive power of the absolute value $\left|\lambda_T(a) - \lambda_{\widetilde{T}}(\tilde{a})\right|$ in the objective function, then the resulting problem can still be solved in a way analogous to that described above.

Theorem 7 *Consider the class of minimization problems obtained from* CLOSEST-TCBL *by replacing its objective function with*

$$\biguplus_a \left|\lambda_T(a) - \lambda_{\widetilde{T}}(\tilde{a})\right|^d, \tag{5}$$

with $\biguplus$ *representing either* max *or* $\sum$*, and with* $d > 0$.

If no blob of N is redundant, then any of these problems can be solved in time $O(2^{2k} \cdot n)$ using $O(k \cdot 2^k \cdot n)$ space, where n is the number of taxa in N and T, and k is the level of N.

Proof In the proof of Theorem 6, replace every occurrence of $|\ldots|$ with $|\ldots|^d$, and – if $\biguplus$ represents a sum – replace every occurrence of max with $\sum$. □

It is worth pointing out that the algorithm described in the proofs above requires storing *all* switchings of a blob that are topologically compatible with the input tree. This is unlike the algorithms shown before, where a number of checks on arc lengths (quite stringent ones in the case of the algorithm for TCBL) ensure that, on realistic instances, the number of switchings stored for a blob with k reticulations will be much smaller than 2^k.

Moreover, again unlike the previous algorithms, the queries at step 3c involve considering *all* switchings stored for a child blob C_i, which is what causes the factor 2^{2k} in the runtime complexity. We note that, for certain objective functions, it might be possible to make this faster (with some algorithmic effort), but in order to achieve the generality necessary for Theorem 7, we have opted for the simple algorithm described above.

5 Discussion

In this paper, we have considered the problem of determining whether a tree is displayed by a phylogenetic network, when branch lengths are available. We have shown that, if the network is permitted to have redundant blobs (i.e. nontrivial biconnected components with only one outgoing arc), then the problem becomes hard when at least one of the following two conditions hold: (1) the level of the network is unbounded (Theorem 1), (2) branch lengths are potentially long (Theorem 2). If neither condition holds (i.e. branch lengths are short and level is bounded) then – even when redundancy is allowed – the problem becomes tractable (Theorem 4). We note that phylogenetic networks with redundant blobs are unlikely to be encountered in practice, as their reconstructability from real data is doubtful [24, 20, 33]. This is relevant because, if redundant blobs are not permitted, the problem becomes fixed-parameter tractable in the level of the network (Theorem 3) *irrespective* of how long the branches are.

Building on our result on networks with no redundant blobs, we have then shown how the proposed strategy can be extended to solve a number of variants of the problem accounting for uncertainty in branch lengths. This includes the case where an interval of possible lengths is provided for each branch of the input tree (Theorem 5), and the case where we want to find – among all trees displayed by the network with the same topology as the input tree T – one that is closest to T, according to a number of measures of discrepancy between branch length assignments (Theorems 6 and 7).

The fixed parameter algorithms we present here have runtimes and storage requirements that grow exponentially in the level of the network. However, in the case of storage, this is a worst-case scenario: in practice, this will depend on the number of "viable" switchings stored for each blob, that is, the switchings that pass all checks on topology and branch lengths. In the case of the algorithm for TCBL (Theorem 3), where strict equalities between arc lengths in T and path lengths in N must be verified, we can expect it to be very rare that multiple switchings will be stored for one blob. Similarly, in the case of the algorithm for RELAXED-TCBL (Theorem 5), when the input intervals are sufficiently small, we can expect the number of stored switchings to be limited. In some particular cases, it might even be possible to find the few viable switchings for a blob, without having to consider all $O(2^k)$ switchings, thus removing this factor from the runtime complexity as well.

The algorithm for TCBL (Algorithm 1) provides a good example of the effect of taking into account branch lengths in the tree containment problems: if all checks on branch lengths are removed, what is left is an algorithm that solves the classic (topology-only) tree containment problem, and also provides all ways to locate the input tree in the network (for each blob, it can produce a list of possible images of the corresponding part of of the input tree). This algorithm may run a little faster than Algorithm 1 (as no queries to child blobs are necessary). However, for a small computational overhead, including branch lengths allows to locate more precisely the displayed trees, and provides more strict answers to the tree containment problem.

References

1. Abbott, R., Albach, D., Ansell, S., Arntzen, J., Baird, S., Bierne, N., Boughman, J., Brelsford, A., Buerkle, C., Buggs, R., et al.: Hybridization and speciation. Journal of Evolutionary Biology **26**(2), 229–246 (2013)
2. Bapteste, E., van Iersel, L., Janke, A., Kelchner, S., Kelk, S., McInerney, J.O., Morrison, D.A., Nakhleh, L., Steel, M., Stougie, L., et al.: Networks: expanding evolutionary thinking. Trends in Genetics **29**(8), 439–441 (2013)
3. Baroni, M., Semple, C., Steel, M.: Hybrids in real time. Systematic Biology **55**(1), 46–56 (2006)
4. Bordewich, M., Tokac, N.: An algorithm for reconstructing ultrametric tree-child networks from inter-taxa distances. Discrete Applied Mathematics (2016). In press.
5. Boto, L.: Horizontal gene transfer in evolution: facts and challenges. Proceedings of the Royal Society B: Biological Sciences **277**(1683), 819–827 (2010)
6. Cardona, G., Llabrés, M., Rosselló, F., Valiente, G.: A distance metric for a class of tree-sibling phylogenetic networks. Bioinformatics **24**(13), 1481–1488 (2008)
7. Chan, H.L., Jansson, J., Lam, T.W., Yiu, S.M.: Reconstructing an ultrametric galled phylogenetic network from a distance matrix. Journal of Bioinformatics and Computational Biology **4**(04), 807–832 (2006)
8. Choy, C., Jansson, J., Sadakane, K., Sung, W.K.: Computing the maximum agreement of phylogenetic networks. Theoretical Computer Science **335**(1), 93–107 (2005)
9. Cordue, P., Linz, S., Semple, C.: Phylogenetic networks that display a tree twice. Bulletin of Mathematical Biology **76**(10), 2664–2679 (2014)
10. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT Press and McGraw-Hill (2001)
11. Doolittle, W.F.: Phylogenetic classification and the Universal Tree. Science **284**, 2124–2128 (1999)
12. Downey, R.G., Fellows, M.R.: Fundamentals of parameterized complexity, vol. 4. Springer (2013)
13. Doyon, J.P., Scornavacca, C., Gorbunov, K.Y., Szöllösi, G.J., Ranwez, V., Berry, V.: An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications, and transfers. In: Proceedings of the Eighth RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG'10), *LNCS*, vol. 6398, pp. 93–108. Springer (2011)
14. Gambette, P., Berry, V., Paul, C.: The structure of level-k phylogenetic networks. In: CPM09, *LNCS*, vol. 5577, pp. 289–300. Springer (2009)
15. Garey, M.R., Johnson, D.S.: Complexity results for multiprocessor scheduling under resource constraints. SIAM Journal on Computing **4**(4), 397–411 (1975)
16. Garey, M.R., Johnson, D.S.: Computers and intractability. W. H. Freeman and Co. (1979). A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences
17. Gramm, J., Nickelsen, A., Tantau, T.: Fixed-parameter algorithms in phylogenetics. The Computer Journal **51**(1), 79–101 (2008)

18. Gusfield, D.: ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks. MIT Press (2014)
19. Hotopp, J.C.D.: Horizontal gene transfer between bacteria and animals. Trends in Genetics **27**(4), 157–163 (2011)
20. Huber, K.T., van Iersel, L., Moulton, V., Wu, T.: How much information is needed to infer reticulate evolutionary histories? Systematic Biology **64**(1), 102–111 (2015)
21. Huson, D.H., Rupp, R., Scornavacca, C.: Phylogenetic networks: concepts, algorithms and applications. Cambridge University Press (2010)
22. Huson, D.H., Scornavacca, C.: A survey of combinatorial methods for phylogenetic networks. Genome Biology and Evolution **3**, 23–35 (2011)
23. van Iersel, L.: Algorithms, haplotypes and phylogenetic networks. Ph.D. thesis, Eindhoven University of Technology (2009)
24. van Iersel, L., Moulton, V.: Trinets encode tree-child and level-2 phylogenetic networks. Journal of Mathematical Biology **68**(7), 1707–1729 (2014)
25. van Iersel, L., Semple, C., Steel, M.: Locating a tree in a phylogenetic network. Information Processing Letters **110**(23) (2010)
26. Jansson, J., Sung, W.K.: Inferring a level-1 phylogenetic network from a dense set of rooted triplets. Theoretical Computer Science **363**(1), 60–68 (2006)
27. Kanj, I.A., Nakhleh, L., Than, C., Xia, G.: Seeing the trees and their branches in the network is hard. Theoretical Computer Science **401**(1), 153–164 (2008)
28. Kubatko, L.S.: Identifying hybridization events in the presence of coalescence via model selection. Systematic Biology **58**(5), 478–488 (2009)
29. Mallet, J.: Hybrid speciation. Nature **446**(7133), 279–283 (2007)
30. Meng, C., Kubatko, L.S.: Detecting hybrid speciation in the presence of incomplete lineage sorting using gene tree incongruence: a model. Theoretical population biology **75**(1), 35–45 (2009)
31. Morrison, D.A.: Introduction to Phylogenetic Networks. RJR Productions (2011)
32. Nolte, A.W., Tautz, D.: Understanding the onset of hybrid speciation. Trends in Genetics **26**(2), 54–58 (2010)
33. Pardi, F., Scornavacca, C.: Reconstructible phylogenetic networks: Do not distinguish the indistinguishable. PLoS Comput Biol **11**(4), e1004,135 (2015)
34. Posada, D., Crandall, K.A., Holmes, E.C.: Recombination in evolutionary genomics. Annual Review of Genetics **36**(1), 75–97 (2002)
35. Rambaut, A., Posada, D., Crandall, K., Holmes, E.: The causes and consequences of HIV evolution. Nature Reviews Genetics **5**(1), 52–61 (2004)
36. Vuilleumier, S., Bonhoeffer, S.: Contribution of recombination to the evolutionary history of hiv. Current Opinion in HIV and AIDS **10**(2), 84–89 (2015)
37. Warnow, T.J.: Tree compatibility and inferring evolutionary history. Journal of Algorithms **16**(3), 388–407 (1994)
38. Yu, Y., Degnan, J.H., Nakhleh, L.: The probability of a gene tree topology within a phylogenetic network with applications to hybridization detection. PLoS Genet **8**(4), e1002,660 (2012)
39. Yu, Y., Dong, J., Liu, K.J., Nakhleh, L.: Maximum likelihood inference of reticulate evolutionary histories. PNAS **111**(46), 16,448–16,453 (2014)
40. Zhaxybayeva, O., Doolittle, W.F.: Lateral gene transfer. Current Biology **21**(7), R242–R246 (2011)

Anatomie, animaux, vocabulaire de la vivisection

Philippe Gambette, Tita Kyriacopoulou, Nadège Lechevrel, Claude Martineau

Anatomie, animaux, vocabulaire de la vivisection

Construire des ressources lexicales pour visualiser une thématique dans un corpus littéraire

Philippe Gambette, Tita Kyriacopoulou,
Nadège Lechevrel, Claude Martineau
UPEM, laboratoire LIGM & ANR/DFG BIOLOGRAPHES, FMSH

Cet article propose une méthodologie d'annotation et de visualisation, en vue de l'analyse, de textes d'un corpus littéraire sur la thématique de l'expérimentation animale. Elle se fonde notamment sur l'extraction du vocabulaire relatif à cette thématique qui concerne plus précisément l'anatomie, les animaux ainsi que l'expérimentation. Pour cela, nous combinons deux outils, Unitex et TreeCloud, afin, d'une part, d'enrichir des ressources linguistiques pour la langue française présentes dans la distribution d'Unitex, et d'autre part de visualiser les thématiques d'intérêt au sein du corpus, au fil du texte, ou de manière synthétique.

Introduction

Le projet de recherche Animalhumanité visait à réunir chercheuses et chercheurs en littérature, sciences du vivant et informatique pour des travaux sur l'expérimentation et la fiction mettant l'animalité au cœur du vivant, en valorisant à la fois les fonds des collections du musée Fragonard et le fonds ancien de la bibliothèque de l'EnvA, École nationale vétérinaire d'Alfort. En raison de la non-disponibilité en version numérique des ouvrages du fonds ancien, nous nous sommes concentrés sur les descriptions des pièces du musée Fragonard référencées sur le site de la Bibliothèque interuniversitaire de santé de Paris[1], ainsi que sur un corpus constitué d'ouvrages, déjà numérisés et

1 http://www.biusante.parisdescartes.fr/histoire/images/index.php?mod=a&orig=enva.

disponibles en mode texte, suggérés par les collègues littéraires impliqués dans le projet. Ce corpus a été traité par deux outils afin d'y mettre en valeur les thématiques d'intérêt du projet : Unitex, un analyseur de corpus fondé sur des ressources linguistiques et TreeCloud un logiciel issu de la textométrie qui produit la visualisation d'un texte sous la forme de nuage arboré[2]

Présentation du corpus

Plusieurs références ont été transmises par les chercheuses et chercheurs en littérature impliqués dans le projet de recherche *AnimalHumanité*, en vue de constituer un corpus textuel sur la thématique de l'expérimentation animale, ou plus précisément de la vivisection. Une première phase du travail a consisté à rechercher des sources numérisées, disponibles au format texte, pour ces références. Des sources variées ont été utilisées : *Wikisource*, le site du *Labex OBVIL*, *Gallica*, *Frantext*, *The Montaigne Project*, les *Bibliothèques virtuelles humanistes*, le *Centre Flaubert*, *Les classiques des sciences sociales*, le *Musée de La Fontaine* et *archive.org*. Finalement, un corpus de 34 textes a été constitué sur le principe de l'« échantillon de convenance »[3], c'est-à-dire en combinant les besoins thématiques avec les contraintes de disponibilité.

Ce corpus, dont la majorité des textes date du XIX^e^ siècle, est disponible sur la page http://eclavit.univ-mlv.fr/animalhumanite. Il s'agit d'un corpus de taille réduite (près de 3 Mo et 500 000 occurrences), relativement hétérogène, en particulier du point de vue de la longueur des textes (certains n'étant que des extraits) ou de la langue utilisée (romans, ouvrages scientifiques, œuvres en français du XVI^e^ siècle non modernisé).

2 Philippe Gambette. *User Manual for TreeCloud,* 2010. http://www.treecloud.org/DOWNLOADS/ManualTreecloud.pdf ; Philippe Gambette, Jean Véronis, "Visualising a Text with a Tree Cloud", IFCS'09 (Proceedings of the International Federation of Classification Societies 2009 Conference), *Studies in Classification, Data Analysis, and Knowledge Organization* n°40, 2010, p. 561-570. https://hal-lirmm.ccsd.cnrs.fr/lirmm-00373643/fr/.

3 Mark Algee-Hewitt, Mark McGurl, "Between Canon and Corpus: Six Perspectives on Twentieth-Century Novels", *Stanford Literary Lab Pamphlet* n° 8, 2015, http://litlab.stanford.edu/LiteraryLabPamphlet8.pdf.

Enrichissement des ressources pour l'annotation des textes par Unitex

Une annotation automatique basée sur des ressources lexicales et des motifs grammaticaux

Unitex est un logiciel libre multilingue et multiplateforme[4] d'analyse de corpus qui fait appel à des ressources linguistiques (dictionnaires et grammaires locales). Il permet en particulier de localiser des *motifs*, c'est-à-dire des mots ou groupes de mots qui correspondent à un patron combinant des contraintes lexicales ou morphosyntaxiques. Ces contraintes peuvent s'exprimer sous forme d'un automate, comme celui illustré en figure 1[5]. Les motifs détectés dans le texte sont alors *annotés*, c'est-à-dire que des balises sont automatiquement ajoutées pour indiquer leur appartenance à une catégorie donnée.

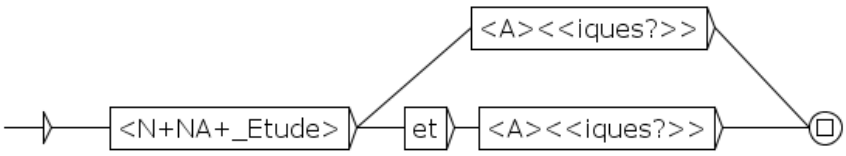


Figure 1 : Automate construit avec Unitex pour reconnaître des expressions du type nom commun, ou groupe nominal, lié à la thématique « études », suivi d'un adjectif se terminant par « iques » ou par « et » suivi d'un adjectif se terminant par « iques ».

À partir de son interface, ce logiciel permet de ne traiter qu'un seul texte à la fois. Pour traiter notre corpus, nous avons donc développé un script en Perl qui appelle directement le cœur du logiciel Unitex et qui permet de produire pour chacun des textes, un texte balisé avec des annotations. Ce texte annoté est ensuite traité par le programme Perl qui génère une page web dans laquelle chaque annotation est surlignée d'une couleur qui indique sa catégorie. En plus du texte annoté, l'outil fournit aussi l'ensemble des motifs

4 Unitex dispose d'un site internet : http://unitexgramlab.org.
5 Sébastien Paumier, De la reconnaissance de formes linguistiques à l'analyse syntaxique, thèse de doctorat, Université de Marne-la-Vallée, 2003, https://hal.archives-ouvertes.fr/tel-01687029.

reconnus, classés par fréquence décroissante ou par catégorie.

Nous avons donc annoté le corpus à l'aide de ce script et obtenu la page web disponible à l'adresse http://eclavit.univ-mlv.fr/animalhumanite à partir des 23 catégories indiquées en figure 2.

Catégorie	Nb d'occ.	Motifs diff.
Outil_Chirurgical	305	25
Médical	497	71
Anomalie	63	8
Biologie	1631	285
Chimie	1294	244
Profession	701	30
Expérimentation	1075	72
Homme_Animal	1089	58
Étude	4675	793
Forme_Verbale	877	296
Personne	1318	391

Catégorie	Nb d'occ.	Motifs diff.
Animal_domestique	533	74
Mammifère	497	123
Oiseau	361	91
Insecte	732	110
Reptile	98	22
Animal	136	37
Pré_Animal	85	17
Cat_Animal	251	74
Partie_Corps	2432	304
Partie_Corps_Animal	367	43
Fluide_Corporel	384	33
Institution	37	3

Figure 2 : Synthèse des 23 catégories recherchées dans le corpus, avec pour chacune le nombre de motifs différents détectés (3204 au total) et d'occurrences reconnues (19438 au total).

Un enrichissement des ressources lexicales utilisées

Nous disposions déjà dans nos dictionnaires de traits de type animal, parties du corps, etc. Cependant, certaines ressources étaient insuffisantes ou inadéquates, ce qui nous a amenés à les compléter à partir de notre corpus, à partir de règles linguistiques ou à partir de la base de données des pièces du musée Fragonard.

En ce qui concerne l'ajout de traits plus précis que ceux déjà présents dans nos ressources, nous avons créé des catégories « animal domestique », « mammifère », « oiseau », « insecte », « reptile » afin d'augmenter la finesse de nos repérages. Un trait « parties du corps animal » a également été ajouté au trait « parties du corps », pour repérer des mots comme « pattes » ou « bec ».

Nous avons également complété certains dictionnaires par des entrées supplémentaires. Un certain nombre de ces nouvelles entrées, sur les parties du corps animal, les animaux ainsi que des anomalies médicales par exemple, proviennent d'une analyse arborée, montrée en figure 3, des titres des pièces du musée Fragonard présents dans une base de données. L'arbre rapproche les mots qui apparaissent fréquemment dans les mêmes titres, et il est coloré en fonction des diverses catégories thématiques proposées (animaux en bleu, parties du corps en rose, anomalies anatomiques en rouge, entités nommées en vert).

La méthode de recherche de motifs caractéristiques de certains traits, implémentée par Unitex, est aussi utilisée pour ajouter de nouvelles entrées à nos ressources lexicales. Par exemple, nous avons constaté lors de la lecture du corpus que plusieurs parties relatives aux descriptions de débats scientifiques font apparaître des mots composés avec des noms suivis d'adjectifs se terminant par « ique », tels que « propriétés mécaniques » ou « sciences biologiques ». Nous avons donc construit l'automate illustré en figure 1 pour repérer ces expressions. À partir de la liste des résultats obtenus, nous avons ajouté à nos dictionnaires ceux qui pouvaient être rattachés à la description d'études scientifiques, en les associant à un trait « étude », et en ajoutant les codes employés par Unitex pour reconnaître automatiquement les formes fléchies, au pluriel, par exemple. Pour de plus amples explications sur ce graphe on pourra consulter les sections 4.3 et 4.7 du manuel d'Unitex[6].

Finalement, plus de 2500 entrées spécifiques ont ainsi été ajoutées à nos dictionnaires.

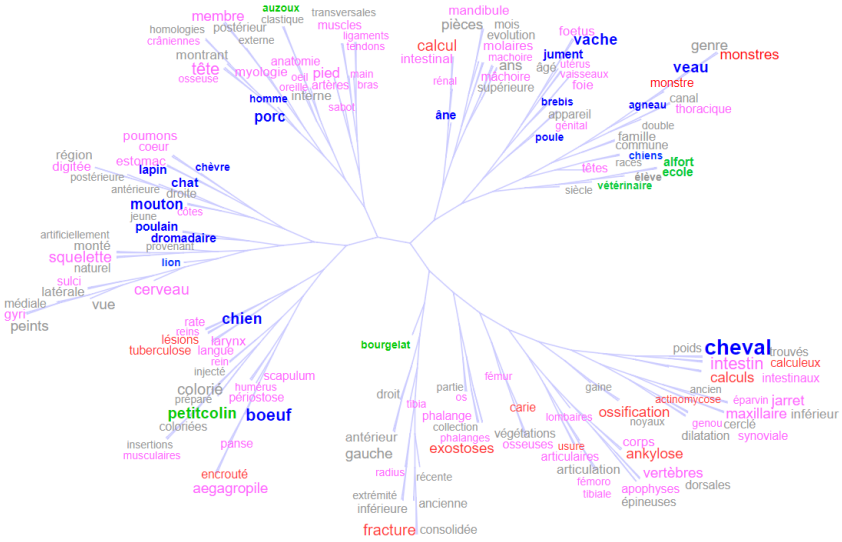


Figure 3 : Nuage arboré des mots présents dans mots dans au moins 15 descriptions de pièces du musée Fragonard de l'EnvA, parmi les 3084 recensées dans la collection EnvA BIU Santé. Disponible sur http://treecloud.univ-mlv.fr/treecloud-linker/fragonard.html en version interactive avec des liens vers le formulaire de recherche dans la collection EnvA.

6 Sébastien Paumier, Claude Martineau, Unitex 3.1, *Manuel d'utilisation*, 2016 http://unitexgramlab.org/releases/3.1/man/Unitex-GramLab-3.1-usermanual-fr.pdf.

Visualisations et analyses du corpus

Notre corpus est assez large et contient plusieurs œuvres où seuls quelques extraits concernent les thématiques du projet *AnimalHumanité*. Nous avons donc commencé par extraire d'un nuage arboré construit sur l'ensemble du corpus la liste des termes les plus fréquents liés à la thématique de l'expérimentation animale : « expérience », « expériences », « supplice », « anatomie », « sang », « sanglants », « émotion », « pitié », « horreur », « peur », « scalpel », « aiguilles », « éther », « phosphore », « poisons », « morte », « mort », « horrible », « barbare », « assassin », « injecter », « enfermer », « souffrir », « mourir ». Nous avons alors observé le voisinage de ces mots à l'aide de l'extraction de concordances (10 mots avant et 10 mots après) et de leur visualisation en nuage arboré.

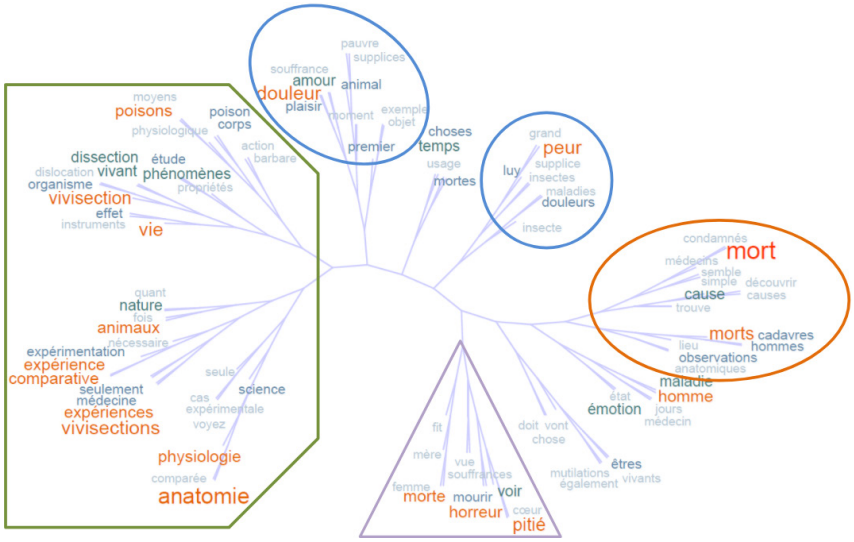


Figure 4 : Nuage arboré des 100 mots les plus fréquents (hors mots vides) dans le voisinage des termes de la catégorie « expérimentations ».

Dans ce nuage arboré montré en figure 4, le sous-arbre de gauche est consacré aux démarches de recherche en sciences du vivant. Il mêle par exemple les disciplines (anatomie, physiologie, médecine), procédés d'étude (vivisection(s), dissection, expérience(s), expérimentation, dislocation), les objets d'étude (organisme, vie, vivant, nature, animaux) et certains moyens utilisés (poison(s), instruments).

Deux sous-arbres en haut de la figure 4 sont consacrés au ressenti de l'animal (douleur(s), peur, supplice(s), souffrance), alors qu'en bas un seul

sous-arbre mêle un lexique lié au féminin (mère, femme, morte) et à des émotions ressenties en observant (vue, voir) les expérimentations (horreur, pitié, cœur). Enfin, un sous-arbre à droite associe à la thématique de la mort (mort(s), cadavres) un vocabulaire essentiellement dénué d'émotion (le mot apparaît plutôt à côté de maladie et homme dans un autre sous-arbre) et plutôt lié à des analyses scientifiques de cadavres (cause(s), observations anatomiques).

Il est aussi possible de se concentrer sur des catégories de mots issues des ressources lexicales, en construisant par exemple avec Unitex les concordances des termes issus de la liste des parties du corps animal. En les visualisant de nouveau à l'aide d'un nuage arboré montré en figure 5, le nombre important d'occurrences de « patte », au pluriel et au singulier, apparaît. D'une façon générale, il y a beaucoup de « pattes » dans *Les Scènes de la vie privée* de Balzac, *Les Sabots de Noël* d'Haraucourt, *L'ennemi des lois* de Barrès et *L'insecte* de Michelet. Dans les textes, les pattes sont surtout celles des chiens, des chats et des chevaux, et le thème des pattes ficelées revient souvent car il est emblématique de la privation de liberté et de la souffrance : être ligoté et retenu pour l'expérimentation.

Les mots « bras » et « pattes » apparaissent à proximité dans l'arbre : un retour au texte montre qu'il peut y avoir des bras et des pattes pour diverses raisons. Ici, dans le texte de Michelet, l'homme et l'animal sont confondus dans une métaphore filée de « *l'insecte géant qu'on appelle cerf-volant, l'un des plus gros de nos climats, masse noire et luisante aux cornes armées de superbes pinces* » où il est tantôt prisonnier, tantôt Roméo[7]. Michelet s'émeut également devant les longs bras d'enfants d'un puceron[8].

L'anthropomorphisme facilite aussi la vulgarisation : Michelet évoque ainsi les dents et la bouche pour décrire les mandibules d'une fourmi[9]. Enfin, la proximité inattendue dans ce corpus des mots « yeux » et « cœur » dans

7 « Il la palpait de ses pattes et de ses bras tremblotants. Il parvint à la retourner, tâtonna (très probablement il ne voyait plus), pour bien s'assurer si elle vivait. Il ne pouvait s'en séparer ; l'on eût juré qu'il avait entrepris, lui mourant, de ressusciter cette morte. » (Jules Michelet. *L'Insecte*, Paris, Librairie Hachette, 1858, http://corpus.biolographes.eu/titre.php?id=185).

8 « Jeté sur le dos, il étalait un ventre très-gros, une très-petite tête informe qui ne semble qu'un suçoir, et remuait toutes ses pattes qu'on eût dit plutôt de longs bras d'enfants. Au total, un être innocent, et qui n'inspire aucune répugnance. » (Jules Michelet, *ibid.*)

9 « Je profitai avec hâte de l'attitude pénible où je tenais ma fourmi : je regardai son visage. Ce qui désoriente le plus et lui donne un aspect étrange, ce sont principalement les dents ou mandibules, placées en dehors de la bouche, et partant l'une de droite, l'autre de gauche, horizontalement, pour se rencontrer ; les nôtres sont verticales. Ces dents en avant menacent et semblent présenter le combat. Cependant, comme nous l'avons dit, elles ont des usages pacifiques et servent aussi de mains. Derrière ces dents apparaissent de petits filets ou palpes, à l'entrée de la bouche. Ce sont en réalité comme de petites mains de la bouche, qui palpent, manient, retournent ce qu'on y apporte. Du front partent les antennes, autres mains, mais du dehors, mobiles à l'excès, sensibles, des mains électriques. » (Jules Michelet, *ibid.*)

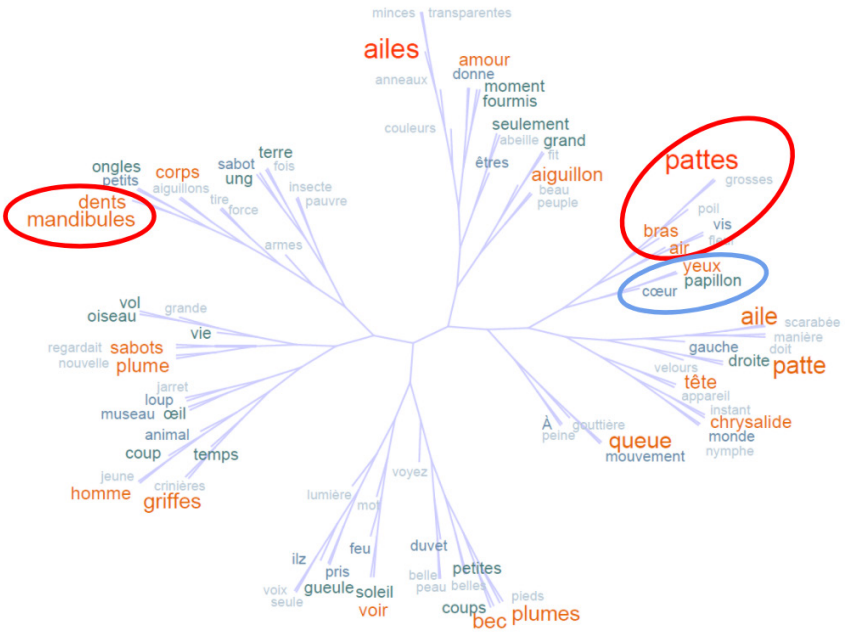


Figure 5 : Nuage arboré des 100 mots les plus fréquents dans les contextes (10 mots avant, 10 mots après) des mots de la catégorie « parties du corps animal ».

la figure 5 s'explique en partie par un extrait où Michelet s'interroge sur l'humanité des insectes[10].

Ainsi, les parties des corps humain et animal sont interchangeables à souhait dans les textes littéraires : c'est le pouvoir effroyable et monstrueux de la vivisection qui fait de l'homme une bête, et révèle l'humanité de l'animal.

Conclusion

La méthode présentée ci-dessus a permis de définir 23 catégories correspondant aux intérêts des personnes impliquées dans ce projet à l'interface de la littérature, des sciences de la vie et de l'informatique. Notons toutefois que les textes antérieurs au XIX^e^ ne sont que très faiblement représentés dans le corpus en raison de leur moindre disponibilité et des difficultés à traiter le français de l'époque avec les ressources linguistiques et les outils informatiques dont nous disposons.

En appliquant des méthodes d'analyse linguistique et statistique à des corpus littéraires, nous facilitons la fouille du corpus en ligne pour les collègues, en fonction de leurs intérêts. Cela permet de mettre en relation des extraits de texte avec des pièces du musée par exemple dans le cadre d'une application mobile. Par ailleurs, les ressources lexicales construites peuvent être utilisées sur d'autres applications ou corpus.

10 « Point de regard dans ses yeux. Nul mouvement sur son masque muet. Sous sa cuirasse de guerre, il demeure impénétrable. Son cœur (car il en a un) bat-il à la manière du mien ? Ses sens sont infiniment subtils, mais sont-ils semblables à mes sens ? Il semble même qu'il en ait à part, d'inconnus, encore sans nom. » (Jules Michelet, *ibid.*)

COUNTING PHYLOGENETIC NETWORKS OF LEVEL 1 AND 2

MATHILDE BOUVEL, PHILIPPE GAMBETTE, AND MAREFATOLLAH MANSOURI

Abstract. Phylogenetic networks generalize phylogenetic trees, and have been introduced in order to describe evolution in the case of transfer of genetic material between coexisting species. There are many classes of phylogenetic networks, which can all be modeled as families of graphs with labeled leaves. In this paper, we focus on rooted and unrooted level-k networks and provide enumeration formulas (exact and asymptotic) for rooted and unrooted level-1 and level-2 phylogenetic networks with a given number of leaves. We also prove that the distribution of some parameters of these networks (such as their number of cycles) are asymptotically normally distributed. These results are obtained by first providing a recursive description (also called combinatorial specification) of our networks, and by next applying classical methods of enumerative, symbolic and analytic combinatorics.

Keywords: phylogenetic networks, level, galled trees, counting, combinatorial specification, generating function, asymptotic normal distribution

Mathematics Subject Classification (2010): 05A15, 05A16, 92D15

1. Introduction

Phylogenetic networks generalize phylogenetic trees introducing reticulation vertices, which have two parents, and represent ancestral species resulting from the transfer of genetic material between coexisting species, through biological processes such as lateral gene transfer, hybridization or recombination. More precisely, binary phylogenetic networks are usually defined as rooted directed acyclic graphs with exactly one root, tree vertices having one parent and 2 children, reticulation vertices having 2 parents and one child and labeled leaves. The leaves are bijectively labeled by a set of taxa, which correspond to currently living species.

As for trees, phylogenetic networks can be rooted or unrooted. Ideally, phylogenetic networks should be rooted, the root representing the common ancestor of all taxa labeling the leaves. But several methods which reconstruct phylogenetic networks, such as combinatorial [HMSW18, vIM18], distance-based [BDM12, WTM14] or parsimony-based methods [PC01, LV14], do not produce inherently rooted networks, but provide unrooted networks where tree vertices and reticulation vertices cannot be distinguished.

An important parameter that allows to measure the complexity of a phylogenetic networks is its level. Phylogenetic trees are actually phylogenetic networks of level 0, and the level of a network N measures "how far from a tree" N is.

The problem of enumerating (rooted or unrooted) trees is a very classical one in enumerative combinatorics. Solving this problem actually led to general methods for enumerating other tree-like structures, where generating functions play a key role. We will review some of these methods in Section 3. These methods have successfully been used by Semple and Steel [SS06] to enumerate two families of phylogenetic networks, namely unicyclic networks and unrooted level-1 networks (also called *galled trees*). Their results include an equation defining implicitly the generating function for unrooted level-1 networks (refined according to two parameters), which yields a closed formula for the number of unrooted level-1 networks with n (labeled) leaves, k cycles, and a total of m edges (also called arcs) across all the cycles. An upper bound on the number of unlabeled galled trees is also provided in [CHT18]. Other counting results have been more recently obtained on other families of phylogenetic networks, for example on so-called normal and tree-child networks [MSW15, FGM19] and on galled networks [GRZ18].

In this paper, we extend the results of Semple and Steel in several ways. First, about unrooted level-1 networks, we provide an asymptotic estimate of the number of such networks with

n (labeled) leaves. We also prove that the two parameters considered by Semple and Steel are asymptotically normally distributed. Second, we consider rooted level-1 networks, whose enumeration does not seem to have been considered so far in the literature. For these networks, we provide a closed formula counting them by number of leaves, together with an asymptotic estimate, and a closed formula for their enumeration refined by two parameters (the number of cycles and number of edges across all the cycles). Moreover, we show that these two parameters are asymptotically normally distributed. Finally, we consider both unrooted and rooted level-2 networks. Similarly, we provide in each case exact and asymptotic formulas for their enumeration, and prove asymptotic normality for some parameters of interest, namely: the number of bridgeless components of strictly positive level, and the number of edges across them. These parameters are a generalization for level-k ($k > 2$) of those considered by Semple and Steel for level-1, in the sense that they quantify how different from a tree these phylogenetic networks are.

The results of this paper rely heavily on analytic combinatorics [FS08]. This framework can also be used to derive uniform random generators (for example with the recursive method [FZC94] or with a Boltzmann sampler [DFLS04]) directly from the specifications of the classes of phylogenetic networks given below. This could be useful for applications in bioinformatics, especially to generate simulated data in order to evaluate the speed or the quality of the output of algorithms dealing with phylogenetic networks.

Table 1 provides an overview of our results, and of where they can be found in the paper.

Type of network	Unrooted, level-1	Rooted, level-1	Unrooted, level-2	Rooted, level-2
Letter $\mathcal{X}$ denoting the class	$\mathcal{G}$ (**g**alled)	$\mathcal{R}$ (**r**ooted)	$\mathcal{U}$ (**u**nrooted)	$\mathcal{L}$ (**l**ast)
Eq. for the EGF $X(z)$	Thm. 4.1 $(*)$	Thm. 5.1	Thm. 6.1	Thm. 7.1
Exact formula for x_n	Thm. 4.1 $(*)$	Prop. 5.2	Prop 8.1	Prop. 8.2
Asymptotic estimate of x_n	Prop. 4.2	Prop. 5.3	Prop. 6.2	Prop. 7.2
Eq. for the multivariate EGF	Eq. (3) $(*)$	Eq. (4)	Eq. (5)	Eq. (6)
Asymptotic normality	Prop. 4.3	Prop. 5.5	Prop. 6.3	Prop. 7.3

TABLE 1. Overview of our main results. EGF means exponential generating function. The results marked with $(*)$ also appear in the work of Semple and Steel [SS06]. In addition, refined enumeration formulas for unrooted and rooted level-1 networks are provided in [SS06, Thm. 4] and Prop. 5.4 respectively. (Although the proof method applies to obtain such formulas for level-2 as well, the computations would however be rather intricate, and the interest *a priori* of the formulas so obtained questionable, hence our choice not to do it.)

For the reader's convenience, we also include in this introduction the beginnings of the enumeration sequences of the four types of networks we consider, as well as their asymptotic behavior – see Table 2. We have added these sequences to the OEIS [OEI19], and we also include in Table 2 their OEIS reference. We also provide in the supplementary material the code in the DOT language, as well as a visualization with GraphViz, of the 15 unrooted level-1 networks on 4 leaves, the 3 rooted level-1 networks on 2 leaves, the 6 unrooted level-2 networks on 3 leaves and the 18 rooted level-2 networks on 2 leaves.

The remainder of the article is organized as follows. First, Section 2 recalls definitions and properties of phylogenetic networks that are important for our purpose. Next, Section 3 reviews some methods of enumerative and analytic combinatorics which we will apply to solve the enumeration of our networks in the following sections. Precisely, Sections 4 and 5 deal with level-1 networks, unrooted and rooted, while Sections 6 and 7 focus on level-2 networks.

n	g_{n-1}	r_n	u_{n-1}	ℓ_n
1	0	1	0	1
2	1	3	1	18
3	2	36	6	1 143
4	15	723	135	120 078
5	192	20 280	5 052	17 643 570
6	3 450	730 755	264 270	3 332 111 850
as $n \to \infty$	$c_1 \approx 0.20748$	$c_1 \approx 0.1339$	$c_1 \approx 0.07695$	$c_1 \approx 0.02931$
$x_n \sim c_1 c_2^n n^{n-1}$ with	$c_2 \approx 1.89004$	$c_2 \approx 2.943$	$c_2 \approx 5.4925$	$c_2 \approx 15.4333$
OEIS reference	A328121	A328122	A333005	A333006

TABLE 2. The numbers of rooted and unrooted level-1 or level-2 networks on n leaves.

2. Some properties of phylogenetic networks

2.1. **Rooted binary phylogenetic networks.** In graph theory, a *cut arc* or *bridge* of a directed graph G is an arc whose deletion disconnects G. A *bridgeless component* of a graph is a maximal induced subgraph of G without cut arcs.

We define a *binary rooted phylogenetic network* N on a set X of leaf labels, for $|X| \geq 2$ as a directed acyclic graph having:

(1) exactly one *root*, that is an in-degree-0 out-degree-2 vertex;
(2) *leaves*, that is in-degree-1 out-degree-0 vertices which are bijectively labeled by elements of X;
(3) *tree vertices*, that is in-degree-1 out-degree-2 vertices;
(4) *reticulation vertices*, that is in-degree-2 out-degree-1 vertices;
and such that
(5) for each bridgeless component B of N, there exist at least two cut arcs of N whose tail[1] belongs to B and whose head does not belong to B.

A binary rooted phylogenetic network N on a singleton x is a single vertex labeled by x.

As illustrated in Fig. 1, a binary rooted phylogenetic network N is said to be *level-k* (or called a *level-k network* for short) if the number of reticulation vertices contained in any bridgeless component of N is less than or equal to k. In a level-1 network N, each bridgeless component B having at least two vertices consists of the union of two directed paths, which start and end at the same vertices, called *source* and *sink* respectively. The source is actually either the root of N, or the head of a cut arc of N, and the sink is the unique reticulation vertex of B. Such bridgeless components are called *cycles*.

Note that variations on the definition of rooted binary phylogenetic networks are around in the literature, and a few comments on our choice of definition are in order. Like in most publications about phylogenetic networks, our definition of binary rooted phylogenetic networks does not allow multiple arcs. As our goal is to study a model of binary phylogenetic networks that could be counted if their number of leaves and level are fixed, condition (5) is necessary to ensure that there are finitely many phylogenetic networks with a given number of leaves and level. Note that this restriction has already appeared in the literature under the name "networks with no redundant biconnected components" [vIM14] or "with no redundant blobs" [GvIK+16]. Indeed, without it, such networks have an unbounded number of vertices: this can be seen by replacing any cut arc of the network by a sequence of networks isomorphic to the one in Fig. 4(2a), which has only one incoming cut-arc and one outgoing cut-arc.

Similarly in some algorithmic-oriented papers about phylogenetic networks, bridgeless components with three vertices and two outgoing arcs are forbidden because the information needed to distinguish those components from simple tree vertices also connected with two outgoing arcs is not available in the input data. In the perspective of counting those objects we do not impose this

[1]The *tail* of an arc is by definition its starting point. Its arrival point is called *head*.

restriction. But it could easily be added to our combinatorial descriptions and formulas below, to be taken into account if needed.

2.2. **Unrooted binary phylogenetic networks.** Now, we extend the latter definition to unrooted phylogenetic networks. A *cut-edge* or *bridge* of an undirected graph G is an edge whose removal disconnects the graph. A *bridgeless component* of a graph G is a maximal induced subgraph of G without cut-edges.

An *unrooted binary phylogenetic network* N on a set X of at least 2 leaf labels is a loopless (undirected) graph whose vertices have either degree 3 (*internal vertices*) or degree 1 (*leaves*), such that its set $L(N)$ of leaves is bijectively labeled by X and such that for each bridgeless component B of N having strictly more than one vertex, the set of cut-edges incident with some vertex of B has size at least 3. An unrooted binary phylogenetic network N on a singleton x is a single vertex labeled by x. An unrooted binary phylogenetic tree is an unrooted binary phylogenetic network with no bridgeless component containing strictly more that one vertex. An unrooted binary phylogenetic network is said to be *level-k* (or called an *unrooted level-k network* for short) if an unrooted binary phylogenetic tree can be obtained by first removing at most k edges per bridgeless component, and then, for each degree-2 vertex, contracting the edge between this vertex and one of its neighbours. We denote by *cycles* the bridgeless components of unrooted level-1 networks having strictly more than one vertex. (Indeed, they are just cycles – of size at least 3 – in the graph-theoretical sense.)

Note that given a rooted level-k network N on n leaves, we can obtain an unrooted binary phylogenetic network N' on $n+1$ leaves with the following *unrooting procedure*: add a vertex adjacent to the root of N, labeled with an extra leaf label (usually denoted #), and ignore all arc directions. Theorem 1 of [GBP12] implies in addition that the network N' so obtained is an unrooted level-k network. This unrooting procedure which consists of building an unrooted level-k network from a rooted level-k network, illustrated in Fig. 1, can be reversed (see Lemma 4.13 of [JJE+18]), although not in a unique fashion. Indeed, given an unrooted level-k network N' on $n+1$ leaves, it is possible to choose any leaf and delete it, making its neighbour become the root ρ of a rooted level-k network N obtained by:

(1) placing the bridgeless component B containing ρ at the top;
(2) orienting downwards all the cut-edges incident with vertices of B;
(3) choosing the tail t of one of these cut arcs as the sink of B;
(4) computing an ρ-t numbering [LEC67] on the vertices of B if there are more than one, that is labeling vertices of B with integers from 1 to the number n_B of vertices of B, such that the labels of ρ and t are respectively 1 and n_B and such that any vertex of B except ρ and t is adjacent both to a vertex with a lower label and a vertex with a higher label;
(5) orienting each edge of B by choosing its vertex with the lower label as the tail; and
(6) moving downwards into the network, recursively applying this procedure on all other bridgeless components.

This correspondence is not one-to-one because of the choices of the leaf which is deleted, and most importantly because of the choices of sinks in step 3 above.

2.3. **Decomposition of rooted and unrooted level-k networks.** For any bridgeless component B with $k_B \leq k$ reticulation vertices of a rooted level-k network N, the directed multi-graph obtained by removing all outgoing arcs, and then contracting each arc from an in-degree-1 out-degree-1 vertex to its child, is called a *level-k_B generator* [vIKK+09, GBP09]. For each $k > 0$, there exists a finite list of *level-k generators* which can be built from level-$(k-1)$ generators [GBP09]. Therefore, depending on the level k_ρ of the bridgeless component B_ρ of N containing its root ρ, N can be decomposed in the following way. It is either:

- a single leaf if $k_\rho = 0$ and ρ has out-degree 0;
- a root ρ being the parent of the root ρ_1 of a rooted level-k network N_1 and of the root ρ_2 of a rooted level-k network N_2 with disjoint sets of leaf labels, if $k_\rho = 0$ and ρ has out-degree 2;

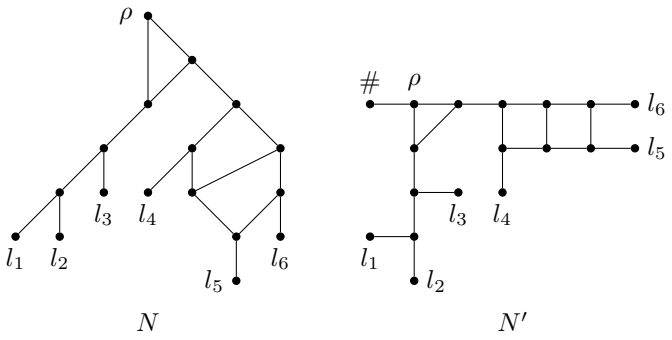


FIGURE 1. A rooted level-2 network N (where all arcs are directed downwards) and the unrooted level-2 network N' obtained by applying the unrooting procedure on N.

- a level-k_ρ generator G_ρ containing the root, with $0 < k_\rho \leq k$, whose arcs are subdivided to create new in-degree-1 out-degree-1 vertices, to which we add a set of cut arcs, whose tails are the out-degree-0 vertices of G_ρ and the newly created in-degree-1 out-degree-1 vertices, and whose heads are roots of rooted level-k networks with disjoint sets of leaf labels.

Similarly, for any bridgeless component B of an unrooted level-k network N, the multi-graph obtained by first removing all cut-edges incident with any vertex of B, then, for each degree-2 vertex, contracting the edge between this vertex and one of its neighbours, is called an *unrooted level-k_B generator* [GBP12, HMW16]. An unrooted level-k_B generator can also be defined as a single vertex for $k_B = 0$, as two vertices linked by a multiple edge for $k_B = 1$, and as a 3-regular bridgeless multi-graph with $2k_B - 2$ vertices for $k_B > 1$ (Lemma 6 of [HMW16]). Therefore, by considering a leaf $l_\#$ of any unrooted level-k network N and the bridgeless component B containing the vertex adjacent to this leaf, depending on the level k_B of B, N can be decomposed in the following way.

- If $k_B = 0$ and B consists of a single vertex of degree 1 in N, then N is just the leaf $l_\#$ adjacent to another leaf.
- If $k_B = 0$ and B is not a single vertex of degree 1 in N, then the leaf $l_\#$ is adjacent to a vertex v of degree 3 in N, such that the other two edges incident to v are cut-edges. N is described by the edge between $l_\#$ and v, plus the two other edges incident with v, which are in turn identified with edges of two unrooted level-k networks N_1 and N_2 with disjoint sets of leaf labels (not containing #) in such a way that v is identified with a leaf $l_{\#1}$ (resp. $l_{\#2}$) of N_1 (resp. N_2), removing the leaf labels of $l_{\#1}$ and $l_{\#2}$ during this identification.
- Otherwise $0 < k_B \leq k$. In this case, N is described by taking a level-k_B generator whose edges are subdivided to insert vertices, and then performing identification of these inserted vertices (in a same flavor as in the previous case). Specifically, one of these inserted vertices is identified with the neighbour of $l_\#$ in N, and all others are identified with leaves of unrooted level-k networks with disjoint sets of leaf labels (not containing #). Again, each leaf that is identified with another vertex looses its label during this identification.

These decompositions of rooted and unrooted level-k networks will be the key to our counting results below.

3. Generating functions: some basics tools and techniques

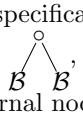
This section summarizes some of the basics on combinatorial classes and their generating functions that we will use in our work. Our presentation follows closely [FS08] (although with much less details), and the reader interested to know more on the topic is referred to [FS08, mainly

Chapters I.5, II.1, II.5, VI.3, VII.3, VII.4]. The reader familiar with the classical tools of analytic combinatorics may safely skip this section.

3.1. **(Univariate) generating functions and counting.** Generally speaking, a *combinatorial class* $\mathcal{C}$ is a set of discrete objects, equipped with a notion of size, such that for every integer n there is a finite number of objects of size n in $\mathcal{C}$. We denote by $\mathcal{C}_n$ the set of objects of size n in $\mathcal{C}$, and by c_n the cardinality of $\mathcal{C}_n$. Specifically in this paper, each combinatorial class we consider is a family of level-k phylogenetic networks, and the size of such a network is its number of leaves.

Objects of size n in $\mathcal{C}$ can be seen as an arrangement (following some rules to be made precise) of n atoms, which are objects of size 1. In our context, these atoms are the leaves of the networks, representing the current species, or *taxa*. When the atoms constituting an object are distinguishable among themselves, the considered combinatorial objects are said to be *labeled*[2]. Because leaves of level-k networks correspond to taxa, our networks are indeed labeled combinatorial objects. Without loss of generality (*i.e.*, up to relabeling), atoms in a labeled object of size n are simply labeled by integers from 1 to n, and we will take this convention in our work.

To a (labeled) combinatorial class $\mathcal{C}$, we can associate its *exponential generating function* $C(z) = \sum_{n\geq 0} c_n \frac{z^n}{n!}$, which is a formal power series in z encapsulating the entire enumeration of $\mathcal{C}$.

A *specification* for a combinatorial class is an unambiguous description of the objects in the class using simpler classes and possibly the class itself. For instance, consider labeled rooted ordered binary trees, and define their size to be the number of their leaves. Such a tree is unambiguously described as being either a leaf or composed of a root to which a left and a right subtree are attached, which are themselves labeled rooted ordered binary trees, with a *consistent relabeling* of their atoms. By this, we mean the following: considering two trees whose atoms are labeled by $\{1,\dots,k\}$ and $\{1,\dots,k'\}$, we can build a tree using the first (resp. second) as left (resp. right) subtree; the atoms of this tree are labeled by $\{1,\dots,k+k'\}$, and need to be such that the relative order between the labels in the left (resp. right) subtree is preserved (and they may be in any such way). This specification for labeled rooted ordered binary trees can be formally written as follows: $\mathcal{B} = \bullet \uplus$  , where $\bullet$ represents a leaf (contributing 1 to the size of the object) and $\circ$ represents an internal node (which contributes 0 to the size).

Specifications describing (labeled) combinatorial classes can be translated into equations satisfied by the corresponding (exponential) generating functions. The precise statement that we refer to is [FS08, Theorem II.1]. The following proposition summarizes the simplest cases of this translation, which we will often use later in this paper.

Proposition 3.1 (Dictionary)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two labeled combinatorial classes. Denote by $A(z)$ and $B(z)$ their respective exponential generating functions. Then the generating function of the class which is the disjoint union of $\mathcal{A}$ and $\mathcal{B}$ (resp. the Cartesian product of $\mathcal{A}$ and $\mathcal{B}$) is $A(z) + B(z)$ (resp. $A(z) \cdot B(z)$). In addition, if $\mathcal{A}$ contains no object of size 0, the class which consists of sequences of objects of $\mathcal{A}$ (*i.e.*, m-tuples of objects of $\mathcal{A}$, for any $m \geq 0$) has generating function $\frac{1}{1-A(z)}$.*

On the previous example of binary trees, it follows from the above proposition that the corresponding generating function satisfies $B(z) = z + B(z)^2$.

The next step is to have access to the enumeration sequence (c_n) of a class $\mathcal{C}$ from an equation satisfied by the generating function $C(z)$ of $\mathcal{C}$. A possible way, especially in the case of tree-like objects, is to appeal to the *Lagrange inversion formula* ([FS08, Theorem A.2]). To state it, we introduce the notation $[z^n]C(z)$ to denote the n-th coefficient of the series $C(z)$; that is to say, writing $C(z) = \sum_{n\geq 0} \frac{c_n}{n!} z^n$, we have $[z^n]C(z) = \frac{c_n}{n!}$, or equivalently $c_n = n! \cdot [z^n]C(z)$.

The Lagrange inversion formula is as follows.

Proposition 3.2 (Lagrange inversion formula)**.** *Assume that a generating function C satisfies an equation of the form $C(z) = z\phi(C(z))$ for $\phi(z) = \sum_{n\geq 0} \phi_n z^n$ a formal power series such that*

[2]Although it is also very classical, the case of *unlabeled* objects (with their corresponding *ordinary* generating functions) will not be useful in our work, and is therefore omitted from our presentation.

$\phi_0 \neq 0$. Then, we have:

$$[z^n]C(z) = \frac{1}{n}[z^{n-1}]\phi(z)^n.$$

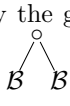
Even though defined as formal power series, it is often useful to consider that generating functions are analytic functions of the complex variable z, in a small disk of convergence around the origin of the complex plane. This sometimes allows to find a closed form for the generating function in its disk of convergence, but not always. Even in this least favorable case, it enables to inherit fundamental results from complex analysis, which can be used for the purpose of enumerating combinatorial objects. In particular, we have in this tool box the *Singular Inversion Theorem* (Theorem VI.6 of [FS08]), which allows to derive asymptotic estimates of the coefficients of generating functions.

Theorem 3.3 (Singular Inversion Theorem)**.** *Let $C(z)$ be a generating function such that $C(0) = 0$, satisfying the equation $C(z) = z\phi(C(z))$ for $\phi(z) = \sum_{n\geq 0}\phi_n z^n$ a power series such that $\phi_0 \neq 0$, all ϕ_n are non-negative real numbers, and $\phi(z) \neq \phi_0 + \phi_1 z$. Denote by R the radius of convergence of ϕ at 0. Assume that ϕ is analytic at 0 (so that $R > 0$), that the* characteristic equation *$\phi(z) - z\phi'(z) = 0$ has a solution $\tau \in (0, R)$ (that is necessarily unique), and that ϕ is aperiodic*[3]*. Then the followings hold:*

- *$\rho = \frac{\tau}{\phi(\tau)}$ is the radius of convergence of C at 0;*
- *near ρ, $C(z) \sim \tau - \sqrt{\frac{2\phi(\tau)}{\phi''(\tau)}}\sqrt{1 - \frac{z}{\rho}}$;*
- *when n grows, $[z^n]C(z) \sim \sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}}\frac{\rho^{-n}}{\sqrt{\pi n^3}}$.*

3.2. **Multivariate generating functions and estimating parameters.** Until now, our generating functions had only a single variable, z, recording the size of the objects we were counting. We now consider *multivariate* generating functions, where additional variables (x, y, ...) record the value of other parameters of our objects. In our cases, we will consider at most two such parameters, which are numbers of certain "substructures" occurring in our objects. Namely, denoting $c_{n,k,m}$ the number of objects of size n in the combinatorial class $\mathcal{C}$ such that the first parameter has value k and the second has value m, the multivariate exponential generating function we consider is $C(z, x, y) = \sum_{n,k,m}\frac{c_{n,k,m}}{n!}z^n x^k y^m$.

To continue our earlier example of binary trees, we could consider one additional parameter, which is the number of internal nodes. (Of course,we are aware that the number of internal nodes is always the number of leaves – *i.e.*, the size – minus one; but we keep this example just to illustrate definitions and tools available.) The coefficient of $z^n x^k$ in the generating function $B(z, x)$ is then the number of binary trees with n leaves and k internal nodes, divided by $n!$.

The "dictionary" translating combinatorial specifications to equations satisfied by the generating function extends to multivariate series, and our earlier specification $\mathcal{B} = \bullet \uplus$  gives $B(z, x) = z + xB(z, x)^2$.

Here again, the Lagrange inversion formula may be used to derive a closed formula for the coefficients $c_{n,k,m}$. Indeed, assuming that our multivariate exponential generating function $C(z, x, y)$ satisfies an equation of the form $C(z, x, y) = z\phi(C(z), x, y)$ for $\phi(z, x, y) = \sum_{n\geq 0}\phi_n(x, y)z^n$ a formal power series such that $\phi_0 \neq 0$, then we have:

$$\frac{c_{n,k,m}}{n!} = [z^n x^k y^m]C(z, x, y) = \frac{1}{n}[z^{n-1}x^k y^m]\phi(z, x, y)^n.$$

Moreover, under some hypotheses, the following theorem (see [Drm09, Theorem 2.23]) allows to prove that the considered parameters are asymptotically normally distributed. The notation used in the statement of this theorem is as follows: if F is a function of several variables, including v, F_v denotes the partial derivative of F with respect to v; as usual, $\mathbb{E}$ and $\mathbb{V}ar$ respectively denote

[3]Aperiodicity is needed only for the third item below. The definition of aperiodicity is omitted from this paper, and can be found in [FS08, Definition IV.5]. A sufficient condition for a power series to be aperiodic (which applies to all examples considered in this paper), is to have $\phi_n > 0$ for all n.

expectation and variance; $\mathcal{N}(0,1)$ is the standard normal distribution; and $\xrightarrow{d}$ denotes convergence in distribution.

Theorem 3.4. *Assume that $C(z,x)$ is a power series that is the (necessarily unique and analytic) solution of the functional equation $C = F(C,z,x)$, where $F(C,z,x)$ satisfies the following assumptions: $F(C,z,x)$ is analytic in C, z and x around 0, $F(C,0,x) = 0$, $F(0,z,x) \neq 0$, and all coefficients $[z^n C^m]F(C,z,1)$ are real and non-negative.*

Assume in addition that the region of convergence of $F(C,z,x)$ is large enough for having non-negative solutions $z = z_0$ and $C = C_0$ of the system of equations

$$\begin{aligned} C &= F(C,z,1) \\ 1 &= F_C(C,z,1) \end{aligned}$$

with $F_z(C_0,z_0,1) \neq 0$ and $F_{CC}(C_0,z_0,1) \neq 0$.

Then, if X_n is a sequence of random variables such that

$$\mathbb{E}x^{X_n} = \frac{[z^n]C(z,x)}{[z^n]C(z,1)},$$

then X_n is asymptotically normally distributed.

More precisely, setting

$$\mu = \frac{F_x}{z_0 F_z}$$

$$\sigma^2 = \mu + \mu^2 + \frac{1}{z_0 F_z^3 F_{CC}}\Big(F_z^2(F_{CC}F_{xx} - F_{Cx}^2) - 2F_zF_x(F_{CC}F_{zx} - F_{Cz}F_{Cx}) + F_x^2(F_{CC}F_{zz} - F_{Cz}^2)\Big)$$

where all partial derivatives are evaluated at the point $(C_0, z_0, 1)$, we have

$$\mathbb{E}X_n = \mu n + O(1) \qquad \text{and} \qquad \mathbb{V}arX_n = \sigma^2 n + O(1)$$

and if $\sigma^2 > 0$ then

$$\frac{X_n - \mathbb{E}X_n}{\sqrt{\mathbb{V}arX_n}} \xrightarrow{d} \mathcal{N}(0,1).$$

3.3. **Implementation and note about computations.** Some of the computations used to obtain the results of this paper were programmed in Maple. A companion Maple document is available from the authors webpage[4].

We also point out to the interested reader that a first version of this article was considering a variant of the model of level-k phylogenetic networks, where multiple (*i.e.* parallel) edges are allowed. The counting results for this alternate model of course differ (starting from level 2), and can be found in [BGM19], again with an associated Maple document[5]. Similarly, these files can easily be used to adapt the computations in case other restrictions are imposed on the structure of level-1 or level-2 phylogenetic networks, for example if *tiny cycles*, defined in [HvIM$^+$17] as bridgeless components with exactly three vertices, are not allowed.

4. Counting unrooted level-1 networks

4.1. **Generating function and exact enumeration formula.** Unrooted level-1 networks (also called unrooted galled trees) have been enumerated in [SS06]. The enumeration does not only consider the number of leaves of the galled trees, but is refined according to two parameters: the number of cycles (*i.e.*, level-1 generators) and the total number of edges which are part of a cycle (that we will call *inner edges*). We only reproduce in Theorem 4.1 a simplified version of the results of [SS06], taking into account the number of leaves only.

[4]at `http://user.math.uzh.ch/bouvel/publications/BouvelGambetteMansouri_Version2_WithoutMultipleEdges.mw`

[5]available at `http://user.math.uzh.ch/bouvel/publications/BouvelGambetteMansouri_Version1_WithMultipleEdges.mw`

Theorem 4.1. *For any $n \geq 0$, let g_n denote the number of unrooted level-1 networks with $(n+1)$ leaves, and denote by $G(z) = \sum_{n\geq 0} g_n \frac{z^n}{n!}$ the corresponding generating function. Then G satisfies the following equation:*

$$G(z) = z + \frac{1}{2}G(z)^2 + \frac{1}{2}\frac{G(z)^2}{1-G(z)},$$

or equivalently

$$G(z) = z\phi(G(z)) \text{ with } \phi(z) = \frac{1}{1-\frac{1}{2}z(1+\frac{1}{1-z})}.$$

Moreover, for any $n \geq 0$, let g_n denote the number of unrooted level-1 networks with $(n+1)$ leaves. We have:

$$g_n = \frac{(2n-2)!}{2^{n-1}(n-1)!} + \sum_{1\leq i\leq k\leq n-1} \frac{(n+i-1)!(n+k-i-2)!}{k!(k-1)!(i-k)!(n-i-1)!}2^{-i}. \quad (1)$$

Notice that even if the formulas seem different, Eq. (1) can be recovered from Theorem 4 of [SS06] by summing over k and m and performing the change of variable $m = n - i + 3k - 1$. The first values of g_n have been included in Table 2.

Proof. We recall the main steps of the proofs of Theorem 4.1 given in [SS06]. To prepare the ground for future proofs, we emphasize their embedding in the context we presented in Section 3.

Since counting rooted objects is far easier that counting unrooted objects, we establish a bijective correspondence between unrooted level-1 networks, and a rooted version of these networks, that we call *pointed* level-1 networks. Pointed level-1 networks on a set of taxa X are simply unrooted level-1 networks on the set of taxa $X \uplus \{\#\}$, where we declare that the leaf labeled by $\{\#\}$ is the "root" of the network. This provides a bijection between unrooted level-1 networks on the set of taxa $X \uplus \{\#\}$ and pointed level-1 networks on X, that have a root labeled by $\{\#\}$. Therefore, there are as many unrooted level-1 networks on the set of taxa $X \uplus \{\#\}$ as pointed level-1 networks on X rooted in a leaf labeled by $\# \notin X$. Hence g_n is the number of pointed level-1 networks with n leaves in addition to the root.

In a pointed level-1 network N (with at least two leaves), we consider the other extremity of the edge to which the root belongs. This vertex may belong to a cycle or not. In the latter case, N is simply described as an unordered pair of two pointed level-1 networks. In the former case, it is described as a non-oriented sequence of at least two pointed level-1 networks. Taking into account the trivial pointed level-1 network with one leaf, a specification for the pointed level-1 networks is therefore the one shown in Fig. 2, where an arrow labeled by *sym* indicates that there is a symmetry w.r.t. the vertical axis to take into account, and the dashed edge corresponds to an edge or a path with internal vertices that are incident with cut-edges, themselves identified with edges of other pointed level-1 networks, the vertex lying on the cycle being identified with a leaf of corresponding network.

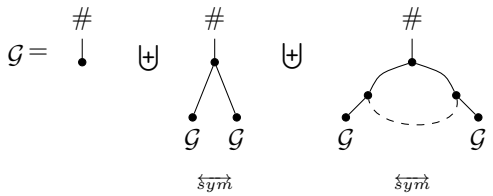


FIGURE 2. The combinatorial specification for unrooted level-1 networks (a.k.a. galled trees).

Thanks to the "dictionary", the generating function therefore satisfies $G(z) = z + \frac{1}{2}G(z)^2 + \frac{1}{2}\frac{G(z)^2}{1-G(z)}$ as claimed by Theorem 4.1. The second statement about $G(z)$ in Theorem 4.1 is obtained by simple algebraic manipulations.

From $G(z) = z\phi(G(z))$, where $\phi(z) = \frac{1}{1-\frac{1}{2}z(1+\frac{1}{1-z})}$, we can apply Lagrange inversion to find g_n. Indeed, $g_n = n![z^n]G(z) = (n-1)![z^{n-1}]\phi(z)^n$.

Recall the following development of $(1-z)^{-n}$, for any $n \geq 1$, which will be used here and several times later on:

$$\left(\frac{1}{1-z}\right)^n = \sum_{i\geq 0} \binom{n+i-1}{i} z^i. \tag{2}$$

Applying this identity twice and the binomial theorem, we get that

$$\begin{aligned}
\phi(z)^n =& \sum_{i\geq 0} \binom{n+i-1}{i} \left(\frac{1}{2}z\left(1+\frac{1}{1-z}\right)\right)^i \\
=& \sum_{i\geq 0} \binom{n+i-1}{i} \left(1 + \sum_{k=1}^{i}\sum_{p\geq 0} \binom{i}{k}\binom{k+p-1}{p} z^p\right) \frac{1}{2^i} z^i \\
=& \sum_{i\geq 0} \binom{n+i-1}{i} \frac{z^i}{2^i} + \sum_{i\geq 0}\sum_{k=1}^{i}\sum_{p\geq 0} \binom{n+i-1}{i}\binom{i}{k}\binom{k+p-1}{p} \frac{z^{i+p}}{2^i}.
\end{aligned}$$

It follows that

$$\begin{aligned}
[z^{n-1}]\phi(z)^n =& \binom{2n-2}{n-1}\frac{1}{2^{n-1}} + \sum_{i=0}^{n-1}\sum_{k=1}^{i} \frac{1}{2^i}\binom{n+i-1}{i}\binom{i}{k}\binom{n+k-i-2}{n-i-1} \\
\text{and } g_n =& \frac{(2n-2)!}{2^{n-1}(n-1)!} + \sum_{1\leq k\leq i\leq n-1} \frac{(n+i-1)!(n+k-i-2)!}{k!(k-1)!(i-k)!(n-i-1)!} 2^{-i}.
\end{aligned}$$

□

4.2. **Asymptotic evaluation.** From Theorem 4.1, we can furthermore derive an asymptotic evaluation of the number g_n of unrooted level-1 networks on $(n+1)$ leaves, using Theorem 3.3.

Proposition 4.2. *The number g_n of unrooted level-1 networks on $(n+1)$ leaves is asymptotically equivalent to $c_1 \cdot c_2^n \cdot n^{n-1}$ for constants c_1 and c_2 such that $c_1 \approx 0.20748$ and $c_2 \approx 1.89004$.*

Proof. Recall that $G(z)$ satisfies $G(z) = z\phi(G(z))$, where $\phi(z) = \frac{1}{1-\frac{1}{2}z(1+\frac{1}{1-z})}$. Equivalently, this can be rewritten as $\phi(z) = \frac{2-2z}{z^2-4z+2}$. So, $\phi(z)$ is a rational fraction, whose pole with smallest absolute value is $2-\sqrt{2} \approx 0.5858$. As such, $\phi(z)$ is analytic at 0, with radius of convergence $R = 2-\sqrt{2}$. Moreover, owing to footnote 3, $\phi(z)$ is aperiodic. Finally, the characteristic equation $\phi(z) - z\phi'(z) = 0$ can be numerically solved (see companion Maple worksheet), showing that it admits a unique solution in the disk of convergence of ϕ, namely $\tau \approx 0.34270$. Therefore, the hypotheses of Theorem 3.3 are all satisfied, and denoting $\rho = \frac{\tau}{\phi(\tau)} \approx 0.19464$, Theorem 3.3 gives:

$$[z^n]G(z) \sim \sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}} \frac{\rho^{-n}}{\sqrt{\pi n^3}}.$$

Using the Stirling estimate of the factorial $n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$, we get:

$$g_n \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}\sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}} \frac{\rho^{-n}}{\sqrt{\pi n^3}} \sim \frac{n^{n-1}}{(e\rho)^n}\sqrt{\frac{\phi(\tau)}{\phi''(\tau)}}.$$

Replacing τ and ρ by their numerical approximations, we obtain the announced result. □

4.3. **Refined enumeration and asymptotic distribution of parameters.** From the specification of pointed level-1 networks seen in the proof of Theorem 4.1, it follows easily, as done in [SS06], that the multivariate generating function $G(z,x,y) = \sum_{n,k,m} \frac{g_{n,k,m}}{n!} z^n x^k y^m$, where $g_{n,k,m}$ is the number of unrooted level-1 networks with $n+1$ leaves, k cycles, and m inner edges, satisfies

$$G(z,x,y) = z + \frac{1}{2}G(z,x,y)^2 + \frac{1}{2}xy^3\frac{G(z,x,y)^2}{1-yG(z,x,y)}. \tag{3}$$

This equation can be rewritten as $G(z,x,y) = z\phi(G(z,x,y),x,y)$ where ϕ is defined by $\phi(z,x,y) = \frac{1}{1-\frac{1}{2}z\left(1+\frac{xy^3}{1-yz}\right)}$. As done in [SS06], we can apply the Lagrange inversion formula to obtain an explicit expression for $g_{n,k,m}$ – see [SS06, Thm. 4].

Using Theorem 3.4, the above equation may also be used to prove that the parameters "number of cycles" and "number of inner edges" are both asymptotically normally distributed.

Proposition 4.3. *Let X_n (resp. Y_n) be the random variable counting the number of cycles (resp. inner edges) in unrooted level-1 networks with $n+1$ leaves. Both X_n and Y_n are asymptotically normally distributed, and more precisely, we have*

$$\mathbb{E}X_n = \mu_X n + O(1), \qquad \mathbb{V}arX_n = \sigma_X^2 n + O(1) \quad \textit{and} \quad \frac{X_n - \mathbb{E}X_n}{\sqrt{\mathbb{V}arX_n}} \xrightarrow{d} \mathcal{N}(0,1),$$

$$\mathbb{E}Y_n = \mu_Y n + O(1), \qquad \mathbb{V}arY_n = \sigma_Y^2 n + O(1) \quad \textit{and} \quad \frac{Y_n - \mathbb{E}Y_n}{\sqrt{\mathbb{V}arY_n}} \xrightarrow{d} \mathcal{N}(0,1),$$

where $\mu_X \approx 0.46$, $\sigma_X^2 \approx 0.18$, $\mu_Y \approx 1.61$ and $\sigma_Y^2 \approx 1.44$.

Proof. Consider first X_n. Defining $G(z,x) := G(z,x,1)$, it holds that

$$\mathbb{E}x^{X_n} = \frac{[z^n]G(z,x)}{[z^n]G(z,1)}.$$

It follows from the equation for $G(z,x,y)$ that $G(z,x) = F(G(z,x),z,x)$, where F is defined by $F(G,z,x) = z\frac{1}{1-\frac{1}{2}G\left(1+\frac{x}{1-G}\right)}$. Being rational, we see immediately that $F(G,z,x)$ is analytic in G, z and x around 0. Moreover, performing the substitution $z = 0$ (resp. $G = 0$) gives $F(G,0,x) = 0$ (resp. $F(0,z,x) = z$, which is not identically 0). Finally, it is readily checked that F satisfies $[z^n G^m]F(G,z,1) \geq 0$ for all n, m (noting for instance that F is obtained using several times the quasi-inverse operator $A \mapsto \frac{1}{1-A}$, which has a combinatorial counterpart, as seen in Proposition 3.1). In addition, we can determine numerically that the system

$$\begin{aligned} G &= F(G,z,1) \\ 1 &= F_G(G,z,1) \end{aligned}$$

admits a solution (G_0, z_0) such that $G_0 \approx 0.3427$ and $z_0 \approx 0.1946$, which satisfies the hypothesis of Theorem 3.4 (see the companion Maple worksheet to determine the solution and to check it satisfies the required hypotheses). The result then follows from Theorem 3.4, and the numerical estimates of μ_X and σ_X^2 are obtained plugging the numerical estimates for G_0 and z_0 into the explicit formulas given by Theorem 3.4 (see again companion Maple worksheet for details). The proof for Y_n follows the exact same steps, considering this time $G(z,y) := G(z,1,y)$ instead, and adjusting the definition of F accordingly. As expected, the solution (G_0, z_0) of the associated system is the same as above. □

Remark 4.4. *In the above proof of Proposition 4.2 (resp. Proposition 4.3), we have provided some details on how Theorem 3.3 (resp. Theorem 3.4) was used and on how its hypotheses were checked. This is omitted in later proofs using Theorem 3.3 (see Propositions 5.3, 6.2 and 7.2) or Theorem 3.4 (see Propositions 4.3, 5.5 and 6.3), since they work following the exact same steps. Note also that all numerical resolutions of equations are done in the companion Maple worksheet.*

5. Counting rooted level-1 networks

5.1. **Combinatorial specification and generating function.** As for unrooted level-1 networks, we start by a combinatorial specification that describes rooted level-1 networks (also called rooted galled trees). Because every cycle in a rooted level-1 network not only has a tree vertex above all other vertices of the cycle, but also a reticulation vertex which is below all other vertices of the cycle, notice that these objects are different from the pointed level-1 networks that we considered in the proof of Theorem 4.1.

Recall that each cycle of a level-1 network has strictly more than one outgoing arc (otherwise there would be an infinite number of level-1 networks on n taxa).

Let us denote by $\mathcal{R}$ the set of rooted level-1 networks. The size of a network of $\mathcal{R}$ is the number of its leaves. Distinguishing on the level (0 or 1) of the bridgeless component containing its root, a network of $\mathcal{R}$ is described in exactly one of the following ways. It may be:

- a single leaf (case $0a$);
- a binary root vertex with two children that are roots of networks of $\mathcal{R}$, whose left-to-right order is irrelevant (case $0b$);
- a cycle containing the root with at least two outgoing cut arcs leading to networks of $\mathcal{R}$. This last possibility splits into two subcases, since the reticulation vertex of the cycle may be a child of the root:
  - a cycle whose reticulation vertex is attached to a network of $\mathcal{R}$, is a child of the root and is the lowest vertex of a path coming from the root, where a sequence of at least one network of $\mathcal{R}$ is attached (case $1a$);
  - a cycle whose reticulation vertex is attached to a network of $\mathcal{R}$, and such that a sequence of at least one network of $\mathcal{R}$ is attached to each path of this cycle, the left-to-right order of these two paths being irrelevant (case $1b$).

The specification for $\mathcal{R}$ is therefore the one given in Fig. 3.

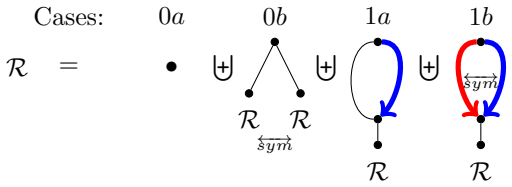


FIGURE 3. The combinatorial specification for rooted level-1 networks. (In this picture, all arcs are directed downwards, the thick arcs each represent a directed path which contains at least one internal vertex incident with a cut arc.)

Denoting r_n the number of rooted level-1 networks on n leaves, and $R(z) = \sum_{n\geq 0} r_n \frac{z^n}{n!}$ the associated exponential generating function, we deduce from the specification that

$$R = z + \frac{1}{2}R^2 + \frac{R^2}{1-R} + \frac{R}{2}\left(\frac{R}{1-R}\right)^2.$$

Unlike for the other generating functions considered in this paper, the above equation for R allows to find a closed formula for R. Indeed, the above equation has four solutions that can be made explicit with the help of a solver. We can further notice that evaluating the generating function $R(z)$ at $z = 0$, we must obtain $R(0) = r_0 = 0$. Among the four candidate solutions for R, we therefore select the only one which has value 0 for $z = 0$ and obtain an explicit form for $R(z)$, given in Theorem 5.1.

Theorem 5.1. *The exponential generating function $R(z)$ of rooted level-1 networks is expressed as*

$$R(z) = -\frac{\sqrt{2}\sqrt{-4(\sqrt{1-8z}-2)z + 9\sqrt{1-8z} - 1}}{4(1-8z)^{\frac{1}{4}}} - \frac{1}{4}\sqrt{1-8z} + \frac{5}{4}$$

within its disk of convergence of radius $\frac{1}{8}$.

5.2. **Exact enumeration formula.** The first terms of the sequence $(r_0, r_1, r_2, \ldots)$ can be read on the Taylor expansion of $R(z)$, and have been collected in Table 2:

$$R(z) = z + 3\frac{z^2}{2!} + 36\frac{z^3}{3!} + 723\frac{z^4}{4!} + 20280\frac{z^5}{5!} + o(z^5).$$

More generally, we have:

Proposition 5.2. *For any* $n \geq 1$*, the number* r_n *of rooted level-1 networks with* n *leaves is given by*

$$\frac{(2n-2)!}{2^{n-1}(n-1)!} + \sum_{\substack{1\leq k\leq i\leq n-1\\ 0\leq p\leq k}} \frac{(n+i-1)!(n+k-i-2)!\quad 2^{p-i}}{(i-k)!(k-p)!p!(n-1-i-k+p)!(2k-p-1)!}.$$

Proof. To obtain a generic formula for r_n, we apply the Lagrange inversion formula, rewriting $R(z)$ as $R(z) = z\phi(R(z))$ where $\phi(z) = \frac{1}{1-\frac{1}{2}z-\frac{z}{1-z}-\frac{1}{2}\left(\frac{z}{1-z}\right)^2}$.

Using twice the usual development of $(1-z)^{-n}$ (for $n \geq 1$) which we recalled in Eq. (2) and twice the binomial theorem, we obtain that

$$\begin{aligned}\phi(z)^n =& \sum_{i\geq 0}\binom{n+i-1}{i}\frac{z^i}{2^i}\\ &+\sum_{i\geq 0}\sum_{k=1}^{i}\sum_{p=0}^{k}\sum_{j\geq 0}\binom{n+i-1}{i}\binom{i}{k}\binom{k}{p}\binom{2k-p+j-1}{j}\frac{z^{i+k-p+j}}{2^{i-p}},\end{aligned}$$

and we deduce that

$$\begin{aligned}r_n &= n![z^n]R(z) = n!\frac{1}{n}[z^{n-1}]\phi(z)^n = (n-1)![z^{n-1}]\phi(z)^n\\ &= \frac{(2n-2)!}{2^{n-1}(n-1)!}\\ &+ \sum_{\substack{1\leq k\leq i\leq n-1\\ 0\leq p\leq k}} \frac{(n+i-1)!(n+k-i-2)!}{(i-k)!(k-p)!p!(n-1-i-k+p)!(2k-p-1)!}2^{p-i}\end{aligned}$$

as announced. □

5.3. **Asymptotic evaluation.** The equation for $R(z)$ also enables us to derive an asymptotic estimate of r_n.

Proposition 5.3. *The number* r_n *of rooted level-1 networks on* n *leaves is asymptotically equivalent to* $c_1 \cdot c_2^n \cdot n^{n-1}$ *for* $c_1 = \frac{\sqrt{34}(\sqrt{17}-1)}{136} \approx 0.1339$ *and* $c_2 = \frac{8}{e} \approx 2.943$.

Proof. Recall that $R(z) = z\phi(R(z))$ where $\phi(z) = \frac{1}{1-\frac{1}{2}z-\frac{z}{1-z}-\frac{1}{2}\left(\frac{z}{1-z}\right)^2}$ so that we can apply the Singular Inversion Theorem. Unlike in the case of unrooted level-1 networks, the solution τ of the characteristic equation $\phi(z) - z\phi'(z) = 0$ to be considered has a nice explicit expression here, and we have $\tau = \frac{5-\sqrt{17}}{4}$. We obtain $\rho = \frac{\tau}{\phi(\tau)} = \frac{1}{8}$ and $\sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}} = \frac{\sqrt{17}(\sqrt{17}-1)}{136}$. Consequently, from Theorem 3.3 we have:

$$[z^n]R(z) \sim \frac{\sqrt{17}(\sqrt{17}-1)}{136}\frac{8^n}{\sqrt{\pi n^3}}.$$

Since $r_n = n![z^n]R(z)$, using the Stirling estimate of the factorial, we finally get:

$$r_n \sim \frac{\sqrt{34}(\sqrt{17}-1)}{136}\left(\frac{8}{e}\right)^n n^{n-1}.$$

□

Notice that with the explicit expression of the generating function $R(z)$ in Theorem 5.1, another way of proving Proposition 5.3 would have been to use the Transfer Theorem (Corollary VI.1 of [FS08]). We do not enter the details of this other method here, but we can check that it gives the same result.

5.4. **Refined enumeration formula.** As in the work of Semple and Steel [SS06], we can refine the enumeration of rooted level-1 networks according to two additional parameters, which are typical of the "level-1" nature of our networks: their number of cycles and their total number of arcs among cycles. To do so, let us introduce the multivariate generating function $R(z,x,y) = \sum \frac{r(n,k,m)}{n!} z^n x^k y^m$, where $r(n,k,m)$ is the number of rooted level-1 networks with n leaves, k cycles and m inner arcs (i.e. the total number of arcs inside those k cycles is m). The specification for $\mathcal{R}$ translates into the following equation for $R = R(z,x,y)$:

$$R = z + \frac{1}{2}R^2 + x\frac{R^2y^3}{1-yR} + xR\frac{1}{2}\left(\frac{Ry^2}{1-yR}\right)^2. \tag{4}$$

The equation can be rewritten as follows:

$$R = z\phi(R,x,y) \text{ where } \phi(z,x,y) = \frac{1}{1-\frac{1}{2}z - x\frac{zy^3}{1-yz} - \frac{1}{2}xy^4\left(\frac{z}{1-yz}\right)^2}.$$

Applying the Lagrange inversion formula again, we have

$$\frac{r(n,k,m)}{n!} = [z^n x^k y^m]R(z,x,y) = \frac{1}{n}[z^{n-1}x^k y^m]\phi(z,x,y)^n,$$

and by the exact same steps of computation as in the proof of Proposition 5.2, we get:

Proposition 5.4. *The number $r(n,k,m)$ of level-1 networks with n leaves, k cycles and m inner arcs (with $k \geq 1$ and $m \geq 1$) is*

$$r(n,k,m) = \sum_{p=0}^{k} \frac{(2n+3k-m-2)!(m-2k-1)!2^{p+m+1-n-3k}}{(n+2k-m-1)!p!(k-p)!(m-4k+p)!(2k-p-1)!}.$$

Notice that from $r_n = r(n,0,0) + \sum_{k=1}^{n-1}\sum_{m=3k}^{n+2k-1} r(n,k,m)$ and the above theorem, we can recover Proposition 5.2 by the change of variable $m = n+3k-i-1$.

5.5. **Asymptotic distribution of parameters.** As we have seen with Proposition 4.3, the equation for the refined generating function does not only give access to the explicit formula of Proposition 5.4 above, but also allows to prove that the two parameters of interest are each asymptotically normally distributed.

Proposition 5.5. *Let X_n (resp. Y_n) be the random variable counting the number of cycles (resp. inner arcs) in rooted level-1 networks with n leaves. Both X_n and Y_n are asymptotically normally distributed, and more precisely, we have*

$$\mathbb{E}X_n = \mu_X n + O(1), \qquad \mathbb{V}arX_n = \sigma_X^2 n + O(1) \quad \text{and} \quad \frac{X_n - \mathbb{E}X_n}{\sqrt{\mathbb{V}arX_n}} \xrightarrow{d} \mathcal{N}(0,1),$$

$$\mathbb{E}Y_n = \mu_Y n + O(1), \qquad \mathbb{V}arY_n = \sigma_Y^2 n + O(1) \quad \text{and} \quad \frac{Y_n - \mathbb{E}Y_n}{\sqrt{\mathbb{V}arY_n}} \xrightarrow{d} \mathcal{N}(0,1),$$

where $\mu_X \approx 0.56$, $\sigma_X^2 \approx 0.18$, $\mu_Y \approx 1.93$ and $\sigma_Y^2 \approx 1.24$.

Proof. Recall that, defining $\phi(z,x,y) = \frac{1}{1-\frac{1}{2}z - x\frac{zy^3}{1-yz} - \frac{1}{2}xy^4\left(\frac{z}{1-yz}\right)^2}$, $R(z,x,y)$ satisfies $R = z\phi(R,x,y)$. We focus first on X_n, setting $y=1$, and we consider $R(z,x) := R(z,x,1)$. It holds that

$$\mathbb{E}x^{X_n} = \frac{[z^n]R(z,x)}{[z^n]R(z,1)}.$$

Defining the function F by $F(R,z,x) = z\phi(R,x,1)$, it also holds that $R(z,x) = F(R(z,x),z,x)$. It is readily checked that F satisfies all hypotheses of Theorem 3.4. Moreover, the system

$$\begin{aligned} R &= F(R,z,1) \\ 1 &= F_R(R,z,1) \end{aligned}$$

admits a solution (R_0, z_0) with $z_0 = 1/8$ and $R_0 \approx 0.2192$, which satisfies the hypothesis of Theorem 3.4. The result and numerical estimates of μ_X and σ_X^2 then follow from Theorem 3.4.

For Y_n instead of X_n, the proof works in the exact same way, considering this time $R(z,y) := R(z,1,y)$ instead, and adjusting the definition of F accordingly. As in the proof of Proposition 4.3, we find the same solution (R_0, z_0) of the associated system, as it should be. □

6. Counting unrooted level-2 networks

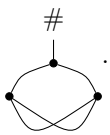
6.1. **Combinatorial specification.** First of all, let us recall that any bridgeless component in an unrooted level-2 network contains at least three vertices incident with a cut-edge (since otherwise there would be an infinite number of such networks with a given number of leaves).

As in the case of level-1 unrooted networks, we consider *pointed* level-2 networks, that are unrooted level-2 networks equipped with a fictitious root, which is a new leaf labeled by the special taxa #. This provides a bijection between unrooted level-2 networks on the set of taxa $X \uplus \{\#\}$ and pointed level-2 networks on X. Therefore, there are as many unrooted level-2 networks of the set of taxa $X \uplus \{\#\}$ as pointed level-2 networks on X rooted in a leaf labeled by $\# \notin X$. Notice that pointed level-2 networks do not correspond to classical rooted level-2 networks. Indeed, every bridgeless component in a pointed level-2 network has a distinguished vertex which could be considered as the equivalent of a root, but no reticulation vertices, whereas it has both in the usual definition of rooted level-2 networks.

Let us denote by $\mathcal{U}$ the set of such pointed level-2 networks, the size of a network of $\mathcal{U}$ being the number of its leaves different from the root. Let u_n be the number of networks of size n in $\mathcal{U}$. The above argument shows that u_n counts the number of unrooted level-2 networks on $(n+1)$ leaves. We introduce $U(z) = \sum_{n\geq 0} u_n \frac{z^n}{n!}$ the associated exponential generating function.

To obtain a combinatorial specification for $\mathcal{U}$, and hence an equation satisfied by $U(z)$, we describe the possible shapes of a network N of $\mathcal{U}$, depending on the level (0, 1 or 2) of the bridgeless component that contains the neighbouring vertex of the fictitious root.

Let v be the neighbor of the fictitious root. Then we have the following cases to consider, depending on the level of the bridgeless component that contains v.

- The first case is that v is a leaf.
- If v does not belong to a cycle nor to a bridgeless component of level 2, then N is described as an unordered pair of two pointed level-2 networks.
- If v belongs to a cycle but not to a bridgeless component of level 2, then N is described as an unoriented sequence of at least two pointed level-2 networks.
  (These first three cases are the same as in Section 5.)
- The last possibility is that v belongs to a bridgeless component of level 2. The underlying level-2 generator, G, is necessarily of the shape 

. In this case, we distinguish many cases in Section 8.1 of the Appendix, depending on whether each edge of the level-2 generator contains exactly one vertex incident with a cut-edge, several or none.

6.2. **Generating function.** The specification is directly translated into the following equation for the generating function U (see the Appendix for details):

$$U = z + \frac{U^2}{2} + \frac{U^2}{2(1-U)} + \frac{U^2}{2(1-U)} + \frac{3}{2}U^2 + \frac{5U^3}{2(1-U)} + \frac{5U^4}{4(1-U)^2} + U^3 + \frac{3U^4}{1-U} + \frac{3U^5}{(1-U)^2} + \frac{U^6}{(1-U)^3} + \frac{U^4}{4} + \frac{U^5}{1-U} + \frac{3U^6}{2(1-U)^2} + \frac{U^7}{(1-U)^3} + \frac{U^8}{4(1-U)^4}.$$

This equation for the generating function allows to derive the first coefficients of the series expansion of $U(z)$, namely:

$$U(z) = z + 3z^2 + \frac{45}{2}z^3 + \frac{421}{2}z^4 + \frac{8809}{4}z^5 + \cdots.$$

The corresponding first values of u_n have been included in Table 2. (Recall indeed that $U(z) = \sum_{n\geq 0} u_n \frac{z^n}{n!}$ and that u_n is the number of unrooted level-2 networks on $(n+1)$ leaves).

The above equation for $U(z)$ can also be rewritten as follows:

Theorem 6.1. *The generating function $U(z)$ satisfies:*

$$U(z) = z\phi(U(z)) \text{ where } \phi(z) = \frac{1}{1 - \frac{3z^5-16z^4+32z^3-30z^2+12z}{4(1-z)^4}}.$$

Proof. This is simply obtained from the above equation for U by algebraic manipulations. □

6.3. **Exact enumeration formula.** To obtain a closed form for u_n, we start from the equation for U given in Theorem 6.1. By the Lagrange inversion formula we obtain that:

$$u_n = n![z^n]U(z) = \frac{n!}{n}[z^{n-1}]\phi^n(z) = (n-1)![z^{n-1}]\phi^n(z),$$

so, to compute the first values of u_n, we can compute the Taylor expansions of $\phi^n(z)$ and get the values in Table 2.

As for the case of level-1 networks, we may also deduce with routine algebra an explicit formula for u_n. This formula being however rather involved, we provide it only in Appendix.

6.4. **Asymptotic evaluation.** From Theorem 6.1, we can furthermore derive an asymptotic evaluation of the number u_n of unrooted level-2 networks on $(n+1)$ leaves, using Theorem 3.3.

Proposition 6.2. *The number u_n of unrooted level-2 networks on $(n+1)$ leaves is asymptotically equivalent to $c_1 \cdot c_2^n \cdot n^{n-1}$ for constants c_1 and c_2 such that $c_1 \approx 0.07695$ and $c_2 \approx 5.4925$.*

Proof. Denoting by $\tau \approx 0.12117$ the unique solution of the characteristic equation $\phi(z) - z\phi'(z) = 0$ in the disk of convergence of ϕ, and $\rho = \frac{\tau}{\phi(\tau)} \approx 0.06698$, we have:

$$[z^n]U(z) \sim \sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}} \frac{\rho^{-n}}{\sqrt{\pi n^3}}.$$

Using the Stirling estimate of the factorial, we get:

$$u_n \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}} \frac{\rho^{-n}}{\sqrt{\pi n^3}} \sim \frac{n^{n-1}}{(e\rho)^n} \sqrt{\frac{\phi(\tau)}{\phi''(\tau)}}.$$

Replacing τ and ρ by their numerical approximations, we get the announced result. □

6.5. **Refined enumeration formula and asymptotic distribution of parameters.** Consider the refined generating function $U(z,x,y)$ for unrooted level-2 networks, where the variable z counts the size as before, the variable x counts the number of bridgeless components of level 1 or 2 (or equivalently, the number of level-1 or level-2 generators in the decomposition of these networks), and the variable y counts the number of inner edges, defined as the total number of edges across all level-1 and level-2 bridgeless components. The specification provided in the Appendix can be refined for these statistics, yielding the following equation for $U := U(z,x,y)$:

$$\begin{aligned} U = z &+ \frac{U^2}{2} + \frac{xy^3U^2}{2(1-yU)} + \frac{xy^6U^2}{2(1-yU)} + \frac{3}{2}xy^6U^2 + \frac{5xy^7U^3}{2(1-yU)} + \frac{5xy^8U^4}{4(1-yU)^2} \\ &+ xy^7U^3 + \frac{3xy^8U^4}{1-yU} + \frac{3xy^9U^5}{(1-yU)^2} + \frac{xy^{10}U^6}{(1-yU)^3} + \frac{xy^8U^4}{4} + \frac{xy^9U^5}{1-yU} \\ &+ \frac{3xy^{10}U^6}{2(1-yU)^2} + \frac{xy^{11}U^7}{(1-yU)^3} + \frac{xy^{12}U^8}{4(1-yU)^4}. \end{aligned} \tag{5}$$

From the above equation, and similarly to Proposition 5.4, it would be possible (although computations and result are not reported in this paper) to derive an explicit formula for the number of unrooted level-2 networks with n leaves, k bridgeless components of level 1 or 2, and m edges across them. Furthermore, some information on the asymptotic behavior of these parameters can be obtained from Eq. (5).

Proposition 6.3. *Let X_n (resp. Y_n) be the random variable counting the number of level-1 or level-2 bridgeless components (resp. the number of edges across them) in unrooted level-2 networks with $n+1$ leaves. Both X_n and Y_n are asymptotically normally distributed, and more precisely, we have*

$$\mathbb{E}X_n = \mu_X n + O(1), \qquad \mathbb{V}arX_n = \sigma_X^2 n + O(1) \quad \textit{and} \quad \frac{X_n - \mathbb{E}X_n}{\sqrt{\mathbb{V}arX_n}} \xrightarrow{d} \mathcal{N}(0,1),$$

$$\mathbb{E}Y_n = \mu_Y n + O(1), \qquad \mathbb{V}arY_n = \sigma_Y^2 n + O(1) \quad \textit{and} \quad \frac{Y_n - \mathbb{E}Y_n}{\sqrt{\mathbb{V}arY_n}} \xrightarrow{d} \mathcal{N}(0,1),$$

where $\mu_X \approx 0.69944$, $\sigma_X^2 \approx 0.16919$, $\mu_Y \approx 4.01349$ and $\sigma_Y^2 \approx 4.68675$.

Proof. Consider first X_n. Defining $U(z,x) := U(z,x,1)$, it holds that

$$\mathbb{E}x^{X_n} = \frac{[z^n]U(z,x)}{[z^n]U(z,1)}.$$

It follows from the equation for $U(z,x,y)$ that $U(z,x) = F(U(z,x),z,x) = z\frac{1}{1-A(U(z,x),z,x)}$ with

$$\begin{aligned} A(U,z,x) = &\frac{U}{2} + \frac{xU}{2(1-U)} + \frac{xU}{2(1-U)} + \frac{3xU}{2} + \frac{5xU^2}{2(1-U)} + \frac{5xU^3}{4(1-U)^2} + xU^2 \\ &+ \frac{3xU^3}{1-U} + \frac{3xU^4}{(1-U)^2} + \frac{xU^5}{(1-U)^3} + \frac{xU^3}{4} + \frac{xU^4}{1-U} + \frac{3xU^5}{2(1-U)^2} \\ &+ \frac{xU^6}{(1-U)^3} + \frac{xU^7}{4(1-U)^4}. \end{aligned}$$

It is readily checked that F satisfies all hypotheses of Theorem 3.4. The system

$$\begin{aligned} U &= F(U,z,1) \\ 1 &= F_U(U,z,1) \end{aligned}$$

admits a solution (U_0, z_0) such that $U_0 \approx 0.1212$ and $z_0 \approx 0.06698$, which satisfies the hypothesis of Theorem 3.4. The result then follows from Theorem 3.4, and the numerical estimates of μ_X and σ_X^2 are obtained plugging the numerical estimates for U_0 and z_0 into the explicit formulas given by Theorem 3.4. The proof for Y_n follows the exact same steps, considering this time $U(z,y) := U(z,1,y)$ instead, and adjusting the definition of F accordingly. Again, as expected, the solution (U_0, z_0) of the associated system is the same as above. □

7. Counting rooted level-2 networks

7.1. **Combinatorial specification and generating function.** To derive a specification for rooted level-2 networks, we distinguish cases depending on the level (0, 1 or 2) of the generator to which the root belongs. The cases corresponding to levels 0 and 1 will be the same as in Section 5. When the root of a rooted level-2 network belongs to a level-2 generator, we have to remember that these generators have one vertex which is above all their other vertices (which is the root of the network) and not just one but two reticulation vertices. As for rooted level-1 networks, it is important to keep in mind that any bridgeless component of level 2 in a rooted level-2 network has at least two outgoing cut arcs (since otherwise there would be an infinite number of such networks with a given number of leaves).

We denote by $\mathcal{L}$ the set of rooted level-2 networks, where the size corresponds to the number of leaves. And we denote by $L(z)$ the corresponding exponential generating function. Distinguishing on the level (0, 1 or 2) of the bridgeless component containing the root, we can see that any network N of $\mathcal{L}$ satisfies exactly one of the following (see Fig. 4).

- N is just a leaf. This contributes z to the generating function (case $0a$).
- The root of N belongs to a bridgeless component of level 0, that is to say it is a binary root vertex. Its children are themselves networks of $\mathcal{L}$ whose left-to-right order is irrelevant. This contributes $\frac{L^2}{2}$ to the generating function (case $0b$).
- The root of N belongs to a bridgeless component of level 1. This case splits into two subcases, just as in Section 5.
  - Either N consists of a cycle whose reticulation vertex is attached to a network of $\mathcal{L}$, is a child of the root and is the lowest vertex of a path from the root where a sequence of at least one network of $\mathcal{L}$ is attached. This contributes $\frac{L^2}{1-L}$ to the generating function (case $1a$).
  - Or N consists of a cycle whose reticulation vertex is attached to a network of $\mathcal{L}$, and a sequence of at least one network of $\mathcal{L}$ is attached to each path of this cycle (case $1b$). This contributes $\frac{L}{2}\left(\frac{L}{1-L}\right)^2$.
- The root of N belongs to a bridgeless component of level 2. The level-2 generators are displayed in Fig. 4, cases $2a$ to $2d$. From these generators, the networks whose root belong to a bridgeless component of level 2 are obtained attaching networks of $\mathcal{L}$ to their reticulation vertex or vertices with out-degree 0, and possibly replacing their arcs with sequences of at least one network of $\mathcal{L}$. Note that in cases $2b$ and $2d$, depending on our choices for such arcs, we may have to cope with horizontal and vertical symmetry. We study these cases in order, and find their contribution to the generating function $L(z)$.

  - We first deal with the case where the level-2 generator to which the root belongs is of type $2a$. This generator has 5 internal arcs, all distinguished from each other by the structure of the generator. A network of $\mathcal{L}$ is attached to its reticulation vertex of out-degree 0. Moreover, recalling that each bridgeless component must be have at least two outgoing cut arcs, at least one of the five internal arcs of the generator must carry a non-empty sequence of networks of $\mathcal{L}$. Therefore, the contribution of case $2a$ to the generating function of $\mathcal{L}$ is

$$L \cdot \sum_{i=1}^{5} \binom{5}{i} \left(\frac{L}{1-L}\right)^i.$$

  - In the case where the level-2 generator to which the root belongs is of type $2b$, we similarly have 5 internal arcs in the generator, at least one of which must be replaced by a non-empty sequence of networks of $\mathcal{L}$. However, the two arcs e and e' are not distinguishable. The contribution to the generating function is therefore more subtle to analyze, and we perform this detailed analysis in the Section 8.2 of the Appendix. The overall contribution of case $2b$ to the generating function of $\mathcal{L}$ is then shown to

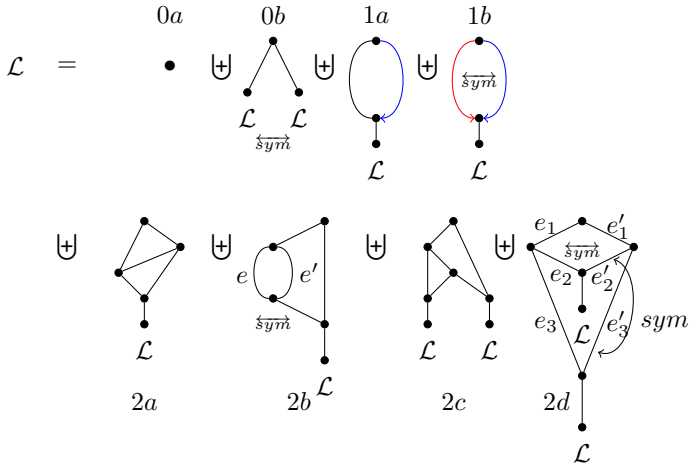


FIGURE 4. The specification of the class $\mathcal{L}$.

be

$$L\frac{L}{1-L}+\frac{7}{2}L\left(\frac{L}{1-L}\right)^2+\frac{9}{2}L\left(\frac{L}{1-L}\right)^3+\frac{5}{2}L\left(\frac{L}{1-L}\right)^4+\frac{1}{2}L\left(\frac{L}{1-L}\right)^5.$$

– We now consider the case where the level-2 generator to which the root belongs is of type $2c$. This generator has 6 internal arcs, all distinguished from each other by the structure of the generator. Moreover, two networks of $\mathcal{L}$ are attached to its reticulation vertices, so that the condition that each bridgeless component must be have at least two outgoing cut arcs is already satisfied. Therefore, all 6 internal arcs of the generator carry possibly empty sequences of networks of $\mathcal{L}$. As a consequence, the contribution of case $2c$ to the generating function of $\mathcal{L}$ is

$$L^2\left(\frac{1}{1-L}\right)^6.$$

– Similarly, when the root belongs to a level-2 generator of type $2d$, the 6 arcs of the generator carry possibly empty sequences of networks of $\mathcal{L}$. However, this generator enjoys both a horizontal symmetry (mapping e_i to e'_i for $i=1,2,3$) and a vertical symmetry (exchanging the indices 2 and 3 and the corresponding pending networks of $\mathcal{L}$). In case all arcs carry empty sequences, the horizontal symmetry is actually the identity, so that only the vertical symmetry applies, yielding a factor $\frac{1}{2}$. Otherwise, both the horizontal and the vertical symmetry need to be taken into account, yielding a factor $\frac{1}{4}$. The total contribution of case $2d$ to the generating function of $\mathcal{L}$ is therefore

$$\frac{1}{2}L^2+\frac{1}{4}L^2\cdot\sum_{i=1}^{6}\binom{6}{i}\left(\frac{L}{1-L}\right)^i.$$

Following this case analysis we obtain an equation characterizing the generating function of $\mathcal{L}$.

Theorem 7.1. *The exponential generating function $L(z)$ of rooted level-2 networks counted by number of leaves satisfies*

$$\begin{aligned}L = z + L^2 + \frac{7L^2}{1-L} + \frac{3L^3}{2(1-L)} + \frac{14L^3}{(1-L)^2} + \frac{15L^4}{4(1-L)^2} + \frac{29L^4}{2(1-L)^3} + \frac{5L^5}{(1-L)^3}\\ + \frac{15L^5}{2(1-L)^4} + \frac{15L^6}{4(1-L)^4} + \frac{3L^6}{2(1-L)^5} + \frac{3L^7}{2(1-L)^5} + \frac{L^2}{(1-L)^6} + \frac{L^8}{4(1-L)^6},\end{aligned}$$

or equivalently

$$L(z) = z\phi(L(z)) \quad \textit{where} \quad \phi(z) = \frac{1}{1 - \frac{36z - 102z^2 + 159z^3 - 148z^4 + 81z^5 - 24z^6 + 3z^7}{4(1-z)^6}}.$$

We therefore obtain the first terms of the series expansion of $L(z)$,

$$L(z) = z + 9z^2 + \frac{381}{2}z^3 + \frac{20013}{4}z^4 + \frac{588119}{4}z^5 + \frac{37023465}{8}z^6 + \cdots,$$

as reported in Table 2.

7.2. **Exact enumeration formula.** As in the previous sections, Theorem 7.1 allows to derive an explicit formula for the number ℓ_n of rooted level-2 phylogenetic networks with n leaves. This complicated formula is given in Appendix.

7.3. **Asymptotic evaluation.** Similarly, from Theorem 7.1, we can also derive the asymptotic behavior of ℓ_n.

Proposition 7.2. *The number ℓ_n of rooted level-2 phylogenetic networks with n leaves behaves asymptotically as*

$$\ell_n \sim c_1 c_2^n n^{n-1},$$

where $c_1 \approx 0.02931$ and $c_2 \approx 15.433$.

Proof. Recall that

$$L(z) = z\phi(L(z)) \quad \textit{where} \quad \phi(z) = \frac{1}{1 - \frac{36z - 102z^2 + 159z^3 - 148z^4 + 81z^5 - 24z^6 + 3z^7}{4(1-z)^6}}.$$

Denoting by $\tau \approx 0.0445$ the unique solution of the characteristic equation $\phi(z) - z\phi'(z) = 0$ in the disk of convergence of ϕ, and $\rho = \frac{\tau}{\phi(\tau)} \approx 0.0238$, the Singular Inversion Theorem gives:

$$[z^n]L(z) \sim \sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}} \frac{\rho^{-n}}{\sqrt{\pi n^3}}.$$

Like before, we get the claimed result from $\ell_n = n![z^n]L(z)$ and the Stirling estimate of the factorial. □

7.4. **Refined enumeration formula and asymptotic distribution of parameters.** Let $L(z, x, y) = \sum_{n,k,m} \ell_{n,k,m} \frac{z^n}{n!} x^k y^m$ be the multivariate generating function counting rooted level-2 networks w.r.t. their number of leaves (variable z), their number of bridgeless components of level 1 or 2 (variable x) and their number of arcs across all these (variable y). From the specification of $\mathcal{L}$ discussed earlier, $L(z, x, y) = L$ is easily seen to satisfy the following equation:

$$\begin{aligned} L = z + \frac{L^2}{2} + x\Big(&\frac{y^6L^2}{2} + (y^3 + 6y^6)\frac{L^2}{1-yL} + \frac{3y^7L^3}{2(1-yL)} + (\tfrac{y^4}{2} + \tfrac{27y^7}{2})\frac{L^3}{(1-yL)^2} \\ &+ \frac{15y^8L^4}{4(1-yL)^2} + \frac{29y^8L^4}{2(1-yL)^3} + \frac{5y^9L^5}{(1-yL)^3} + \frac{15y^9L^5}{2(1-yL)^4} + \frac{15y^{10}L^6}{4(1-yL)^4} \\ &+ \frac{3y^{10}L^6}{2(1-yL)^5} + \frac{3y^{11}L^7}{2(1-yL)^5} + \frac{y^6L^2}{(1-yL)^6} + \frac{y^{12}L^8}{4(1-yL)^6}\Big). \end{aligned} \tag{6}$$

From the above equation, an explicit formula for $\ell_{n,k,m}$ could routinely be derived, as in Proposition 5.4, although the computations are more involved. We decided not to report this formula here. Eq. (6) also allows to study the asymptotic behavior of the considered parameters.

Proposition 7.3. *Let X_n (resp. Y_n) be the random variable counting the number of bridgeless components of level 1 or 2 (resp. the number of edges across them) in rooted level-2 networks with n leaves. Both X_n and Y_n are asymptotically normally distributed, and more precisely, we have*

$$\mathbb{E}X_n = \mu_X n + O(1), \qquad \mathbb{V}arX_n = \sigma_X^2 n + O(1)$$
$$\mathbb{E}Y_n = \mu_Y n + O(1), \qquad \mathbb{V}arY_n = \sigma_Y^2 n + O(1)$$

where $\mu_X \approx 0.8243$, $\sigma_X^2 \approx 0.1232$, $\mu_Y \approx 4.8133$ and $\sigma_Y^2 \approx 3.5523$.

Proof. To prove the result for X_n (resp. Y_n), we specialize Eq. (6) for $y = 1$ (resp. $x = 1$) and rewrite it as $L(z, x, 1) = F(L(z, x, 1), z, x)$ for some explicit function F (resp. $L(z, 1, y) = F(L(z, 1, y), z, y)$, for an explicit different F). It is readily checked that F satisfies all hypotheses of Theorem 3.4, as well as the solutions (L_0, z_0) of the system

$$\begin{aligned} L &= F(L, z, 1) \\ 1 &= F_L(L, z, 1) \end{aligned}$$

whose approximate values are $L_0 \approx 0.04447$ and $z_0 \approx 0.02384$. The result then follows from Theorem 3.4, and the numerical estimates of μ_X and σ_X^2 (resp. μ_Y and σ_Y^2) are obtained plugging the numerical estimates for L_0 and z_0 into the explicit formulas given by Theorem 3.4. □

Acknowledgments

This work was supported by a "junior guest" grant by the LABRI and bilateral Austrian-Taiwanese project FWF-MOST, grants I 2309-N35 (FWF) and MOST-104-2923- M-009-006-MY3 (MOST). We thank Carine Pivoteau for her insights about random generation of combinatorial structures as well as two anonymous reviewers for their useful comments.

References

[BDM12] Alix Boc, Alpha B. Diallo, and Vladimir Makarenkov. T-rex: a web server for inferring, validating and visualizing phylogenetic trees and networks. *Nucleic Acids Research*, 40(W1):W573–W579, 2012.

[BGM19] Mathilde Bouvel, Philippe Gambette, and Marefatollah Mansouri. Counting phylogenetic networks of level 1 and 2. Technical report, Arxiv Preprint, Version 2, 2019.

[CHT18] Kuang-Yu Chang, Wing-Kai Hon, and Sharma V. Thankachan. Compact encoding for galled-trees and its applications. In *2018 Data Compression Conference*, pages 297–306, 2018.

[DFLS04] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, 13:577625, 2004.

[Drm09] Michael Drmota. *Random Trees*. Springer, 2009.

[FGM19] Michael Fuchs, Bernhard Gittenberger, and Marefatollah Mansouri. Counting phylogenetic networks with few reticulation vertices: Tree-child and normal networks. *Australasian Journal of Combinatorics*, 73(2):385–423, 2019.

[FS08] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.

[FZC94] Philippe Flajolet, Paul Zimmermann, and Bernard Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132(1-2):1–35, 1994.

[GBP09] Philippe Gambette, Vincent Berry, and Christophe Paul. The structure of level-k phylogenetic networks. In *Twentieth Annual Symposium on Combinatorial Pattern Matching (CPM'09)*, volume 5577 of *Lecture Notes in Computer Science*, pages 289–300. Springer, 2009.

[GBP12] Philippe Gambette, Vincent Berry, and Christophe Paul. Quartets and unrooted phylogenetic networks. *Journal of Bioinformatics and Computational Biology*, 10(4):1250004.1–1250004.23, 2012.

[GRZ18] Andreas DM Gunawan, Jeyaram Rathin, and Louxin Zhang. Counting and enumerating galled networks, 2018. arXiv manuscript, `https://arxiv.org/abs/1812.08569`.

[GvIK+16] Philippe Gambette, Leo van Iersel, Steven Kelk, Fabio Pardi, and Celine Scornavacca. Do branch lengths help to locate a tree in a phylogenetic network? *Bulletin of Mathematical Biology*, 78(9):1773–1795, 2016.

[HMSW18] Katharina Huber, Vincent Moulton, Charles Semple, and Taoyang Wu. Quarnet inference rules for level-1 networks. *Bulletin of Mathematical Biology*, 80:2137–2153, 2018.

[HMW16] Katharina Huber, Vincent Moulton, and Taoyang Wu. Transforming phylogenetic networks: Moving beyond tree space. *Journal of Theoretical Biology*, 404:30–39, 2016.

[HvIM+17] Katharina Huber, Leo van Iersel, Vincent Moulton, Celine Scornavacca, and Taoyang Wu. Reconstructing phylogenetic level-1 networks from nondense binet and trinet sets. *Algorithmica*, 77(1):173–200, 2017.

[JJE+18] Remie Janssen, Mark Jones, Pter L. Erdös, Leo van Iersel, and Celine Scornavacca. Exploring the tiers of rooted phylogenetic network space using tail moves. *Bulletin of Mathematical Biology*, 80:21772208, 2018.

[LEC67] Abraham Lempel, Shimon Even, and Israel Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs: International Symposium*, pages 215–232, 1967.

[LV14] Anthony Labarre and Sicco Verwer. Merging partially labelled trees: hardness and a declarative programming solution. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 11(2):389–397, 2014.

[MSW15] Colin McDiarmid, Charles Semple, and Dominic Welsh. Counting phylogenetic networks. *Annals of Combinatorics*, 19(1):205–224, 2015.

[OEI19] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. 2019. `http://oeis.org`.

[PC01] David Posada and Keith A. Crandall. Intraspecific gene genealogies: trees grafting into networks. *TEE*, 16(1):37–45, 2001.

[SS06] Charles Semple and Mike Steel. Unicyclic networks: compatibility and enumeration. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 3:398–401, 2006.

[vIKK+09] Leo van Iersel, Judith Keijsper, Steven Kelk, Leen Stougie, Ferry Hagen, and Teun Boekhout. Constructing level-2 phylogenetic networks from triplets. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 6(4):667–681, 2009.

[vIM14] Leo van Iersel and Vincent Moulton. Trinets encode tree-child and level-2 phylogenetic networks. *Journal of Mathematical Biology*, 68(7):1707–1729, 2014.

[vIM18] Leo van Iersel and Vincent Moulton. Leaf-reconstructibility of phylogenetic networks. *SIAM Journal on Discrete Mathematics*, 32:2047–2066, 2018.

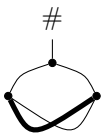
[WTM14] Matthieu Willems, Nadia Tahiri, and Vladimir Makarenkov. A new efficient algorithm for inferring explicit hybridization networks following the neighbor-joining principle. *Journal of Bioinformatics and Computational Biology*, 12(5), 2014.

8. Appendix

8.1. **Case analysis for unrooted level-2 generators.** In the pictures below, we use thick lines to represent paths containing at least 2 internal nodes incident with a cut-edge which is incident with another pointed unrooted level-2 network. We use # to represent the fictitious root in the pointed network, v to denote its neighbour, and $\mathcal{U}$ to represent any pointed network.

8.1.1. *Case 1: One edge with an attached network.* One edge of the generator carries a sequence of at least two incident cut-edges. Because multiple edges are not allowed, it cannot be one of the two edges incident to v. So, it can be only one of the two edges not incident to v (which are not distinguished). The sequence is unoriented, because of symmetry, explaining the factor $\frac{1}{2}$ below.

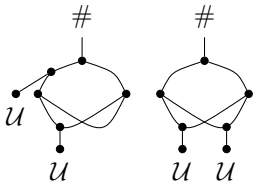
$$\frac{U^2}{2(1-U)}$$



8.1.2. *Case 2: Two edges with attached networks.*

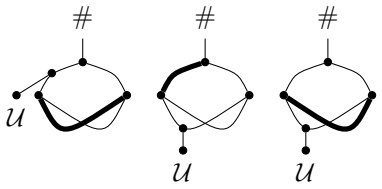
Case 2A - Two edges of the generator carry exactly one incident cut-edge. Since multiple edges are not allowed, it can either be one edge incident to v and one not, or both edges not incident to v. In the latter case, the two edges should not be distinguished, hence the factor $\frac{1}{2}$.

$$U^2 + \frac{U^2}{2} = \frac{3}{2}U^2$$



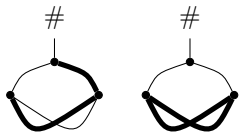
Case 2B - One edge of the generator carries a single incident cut-edge and another edge carries a sequence of at least two incident cut-edges. Again, these cannot be the two edges incident to v. The only case where symmetries need to be taken care of is when the two edges are those not incident to v: in this case, the sequence is not oriented, hence the factor $\frac{1}{2}$. In all other cases, the orientation of the sequence is determined by the presence of the fictitious root or the outgoing arc from the other edge with and attached network.

$$\frac{U^3}{1-U} + \frac{U^3}{1-U} + \frac{U^3}{2(1-U)} = \frac{5U^3}{2(1-U)}$$



Case 2C - Two edges of the generator (but not the two incident to v, as before) carry a sequence of at least two incident cut-edges. If one arc is incident to v and the other not, then both sequences are oriented and there is no symmetry factor. If the two arcs are those not incident to v, then the two sequences they carry can be seen as an unordered pair of oriented sequences, seen up to symmetry w.r.t. the vertical axis. This yields a factor $\frac{1}{2}$ since the pair is unordered, and another factor $\frac{1}{2}$ to account for the symmetry w.r.t. the vertical axis.

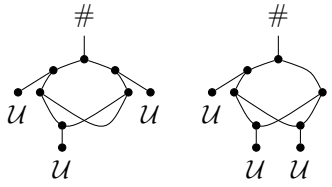
$$\frac{U^4}{(1-U)^2} + \frac{U^4}{4(1-U)^2} = \frac{5U^4}{4(1-U)^2}$$



8.1.3. *Case 3: Three edges with attached networks.*

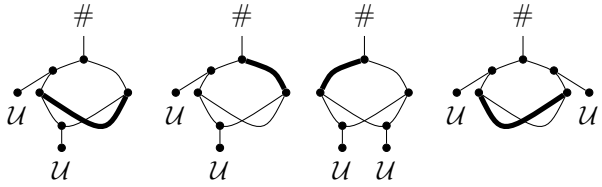
Case 3A - Three edges of the generator carry exactly one incident cut-edge. The unused edge can either be incident with v or not. In both cases, we have a factor $\frac{1}{2}$ because of symmetry.

$$\frac{U^3}{2} + \frac{U^3}{2} = U^3$$



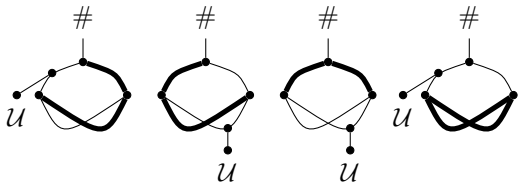
Case 3B - Two edges of the generator carry a single incident cut-edge and one carries a sequence of at least two incident cut-edges. The only cases where a symmetry comes into play here are when the edges carrying a single incident cut-edge are either the two edges incident to v or the two edges not incident to v. This yield the factor $\frac{1}{2}$ in these two cases. Moreover, all sequences are oriented, because of the presence of the fictitious root or the single incident cut-edges.

$$\frac{U^4}{1-U} + \frac{U^4}{1-U} + \frac{U^4}{2(1-U)} + \frac{U^4}{2(1-U)} = \frac{3U^4}{1-U}$$



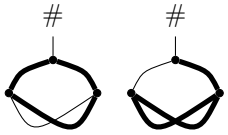
Case 3C - One edge of the generator carries a single incident cut-edge and two edges carry a sequence of at least two incident cut-edges. Similarly to the previous case, we obtain a factor $\frac{1}{2}$ for symmetry reasons when the two edges carrying sequences are either the two edges incident to v or the two edges not incident to v. Moreover, all sequences are oriented, because of the presence of the fictitious root or the single incident cut-edge.

$$\frac{U^5}{(1-U)^2} + \frac{U^5}{(1-U)^2} + \frac{U^5}{2(1-U)^2} + \frac{U^5}{2(1-U)^2} = \frac{3U^5}{(1-U)^2}$$



Case 3D - Three edges of the generator carry a sequence of at least two incident cut-edges. In both cases, we have a factor $\frac{1}{2}$ for symmetry reason, but all sequences are oriented by the presence of the fictitious root, or of the sequence on the edge(s) incident to v.

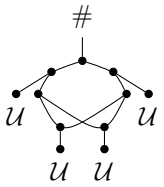
$$\frac{U^6}{2(1-U)^3} + \frac{U^6}{2(1-U)^3} = \frac{U^6}{(1-U)^3}$$



8.1.4. *Case 4: Four edges with attached networks.*

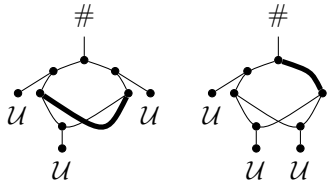
Case 4A - The four edges of the generator each carry exactly one incident cut-edge. In this case, the two edges incident to v can be exchanged without modifying the network, and the same holds for the two edges not incident to v. This yields a factor $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ due to symmetries.

$$\frac{U^4}{4}$$



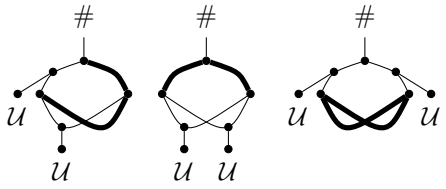
Case 4B - Three edges of the generator carry a single incident cut-edge and the fourth one carries a sequence of at least two incident cut-edges. If this fourth edge is one incident to v, then the sequence it carries is oriented by the presence of the fictitious root, but the two arcs pending on the edges not incident to v are symmetric, hence a factor $\frac{1}{2}$. If on the contrary the edge carrying the sequence is not incident to v, then the sequence is also oriented, this time because of the arcs attached to the edges incident to v. Moreover, the picture has a symmetry w.r.t. the vertical axis, hence a factor $\frac{1}{2}$.

$$\frac{U^5}{2(1-U)} + \frac{U^5}{2(1-U)} = \frac{U^5}{1-U}$$



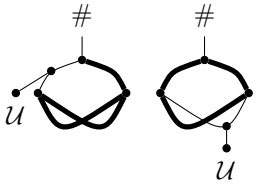
Case 4C - Two edges carry a single incident cut-edge and the two others carry a sequence of at least two incident cut-edges. In all cases, the sequences are oriented, by the presence of either the fictitious root or of the single arcs attached to edges. If the edges carrying sequences are one incident to v and the other not incident to v, all edges are in addition distinguished from each other. In the other two cases, both edges incident to v form an unordered pair, as well as the two edges not incident to v. In each case, we therefore have a factor $\frac{1}{4}$.

$$\frac{U^6}{(1-U)^2} + \frac{U^6}{4(1-U)^2} + \frac{U^6}{4(1-U)^2} = \frac{3U^6}{2(1-U)^2}$$



Case 4D - One edge of the generator carries a single incident cut-edge and three edges carry a sequence of at least two incident cut-edges. As in the previous case, all sequences are oriented. However, if the two edges incident to v carry a sequence, the picture has a symmetry w.r.t. the vertical axis, hence a factor $\frac{1}{2}$. If on the contrary the two edges not incident to v carry a sequence, these two edges are indistinguishable, hence a factor $\frac{1}{2}$ also in this case.

$$\frac{U^7}{2(1-U)^3} + \frac{U^7}{2(1-U)^3} = \frac{U^7}{(1-U)^3}$$



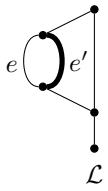
Case 4E - All four edges of the generator carry a sequence of at least two incident cut-edges. Then all sequences are oriented, but the two edges not incident to v are indistinguishable. The picture has in addition a symmetry w.r.t. the vertical axis. This yields a factor $\frac{1}{4}$.

$$\frac{U^8}{4(1-U)^4}$$

8.2. **Case analysis for the rooted level-2 generator 2b.** In the pictures below, we use thick lines to represent paths containing at least one internal node incident with a cut arc which is incident with the root of another rooted level-2 network. All arcs are directed downwards. We use $\mathcal{L}$ to represented any rooted level-2 network.

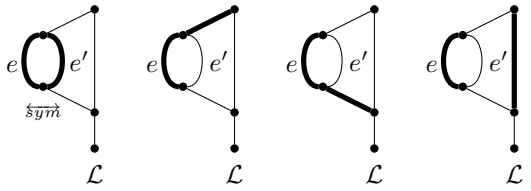
8.2.1. *Case 1:* Only one arc of the generator carries a sequence of at least one outgoing arc. This arc can only be e or e' (and these cases are indistinguishable), since otherwise the network would contain multiple arcs, and this is not allowed.

$$L\frac{L}{1-L}$$



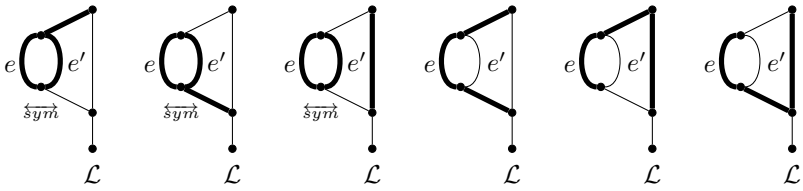
8.2.2. *Case 2:* Exactly two arcs of the generator carry a sequence of at least one outgoing arc. To avoid multiple arcs, either these two arcs are e and e' (and those two arcs are symmetric, hence the factor $\frac{1}{2}$), or one of them is e or e' (which are not distinguished) and the other arc is chosen among the three arcs different from e and e'.

$$\frac{1}{2}L\left(\frac{L}{1-L}\right)^2 + 3L\left(\frac{L}{1-L}\right)^2 = \frac{7}{2}L\left(\frac{L}{1-L}\right)^2$$



8.2.3. *Case 3:* Exactly three arcs of the generator carry a sequence of at least one outgoing arc. Here, there are two possibilities. Either both e and e' are among those three arcs (and those two arcs are symmetric, hence the factor $\frac{1}{2}$). Or, to avoid multiple arcs, we must choose one of e and e' (which are not distinguished from each other), and two additional arcs among the three remaining arcs.

$$\frac{3}{2}L\left(\frac{L}{1-L}\right)^3 + 3L\left(\frac{L}{1-L}\right)^3 = \frac{9}{2}L\left(\frac{L}{1-L}\right)^3$$

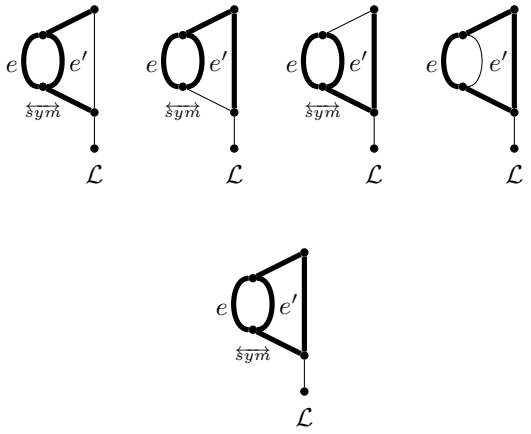


8.2.4. *Case 4:* Exactly four arcs of the generator carry a sequence of at least one outgoing arc. Either both e and e' are among those four arcs (and those two arcs are symmetric, hence the factor $\frac{1}{2}$), so the last two are chosen among the three other arcs of the generator. Or we choose the three arcs of the generator other than e and e', and e (which is undistinguishable from e').

$$\frac{\binom{3}{2}}{2}L\left(\frac{L}{1-L}\right)^4 + L\left(\frac{L}{1-L}\right)^4 = \frac{5}{2}L\left(\frac{L}{1-L}\right)^4$$

8.2.5. *Case 5:* All five arcs of the generator carry a sequence of at least one outgoing arc. The fact that e and e' are symmetric explains the factor $\frac{1}{2}$.

$$\frac{1}{2}L\left(\frac{L}{1-L}\right)^5.$$



8.3. **Exact enumeration formulas.**

8.3.1. *Unrooted level-2 networks.*

Proposition 8.1. *For any $n \geq 1$, the number u_n of unrooted level-2 phylogenetic networks with $(n+1)$ leaves is given by*

$$u_n = (n-1)! \sum_{\substack{0\leq s\leq q\leq p\leq k\leq i\leq n-1\\ j=n-1-i-k-p-q-s\geq 0\\ i\neq 0}} \begin{array}{l}\binom{n+i-1}{i}\binom{4i+j-1}{j}\binom{i}{k}\binom{k}{p}\binom{p}{q}\binom{q}{s}\\ \times(3)^i\left(\frac{-15}{6}\right)^k\left(-\frac{16}{15}\right)^p\left(-\frac{1}{2}\right)^q\left(-\frac{3}{16}\right)^s.\end{array}$$

Sketch. Recall that $U(z) = z\phi(U(z))$ with $\phi(z) = \frac{1}{1-\frac{3z^5-16z^4+32z^3-30z^2+12z}{4(1-z)^4}}$. Using first the classical development of $(1-z)^{-n}$ in series (see Eq. (2)), and then the binomial theorem, we have

$$\begin{aligned}\phi(z)^n =& \sum_{i\geq 0}\binom{n+i-1}{i}\left(\frac{12z}{4(1-z)^4}+\frac{-30z^2+32z^3-16z^4+3z^5}{4(1-z)^4}\right)^i\\ =& \sum_{i\geq 0}\sum_{k=0}^{i}\binom{n+i-1}{i}\binom{i}{k}\left(\frac{12z}{4(1-z)^4}\right)^{i-k}\left(\frac{-30z^2+32z^3-16z^4+3z^5}{4(1-z)^4}\right)^k.\end{aligned}$$

We continue applying the binomial theorem inside the above formula, isolating each time the term with the lowest degree in the numerator (that is, first $\frac{-30z^2}{4(1-z)^4}$, second $\frac{32z^3}{4(1-z)^4}$, ...). This yields

$$\begin{aligned}\phi(z)^n =& \sum_{i\geq 0}\sum_{k=0}^{i}\sum_{p=0}^{k}\sum_{q=0}^{p}\sum_{s=0}^{q}\binom{n+i-1}{i}\binom{i}{k}\binom{k}{p}\binom{p}{q}\binom{q}{s}\\ &\left(\frac{12z}{4(1-z)^4}\right)^{i-k}\left(\frac{-30z^2}{4(1-z)^4}\right)^{k-p}\left(\frac{32z^3}{4(1-z)^4}\right)^{p-q}\left(\frac{-16z^4}{4(1-z)^4}\right)^{q-s}\left(\frac{3z^5}{4(1-z)^4}\right)^{s}\\ =& \sum_{i\geq 0}\sum_{k=0}^{i}\sum_{p=0}^{k}\sum_{q=0}^{p}\sum_{s=0}^{q}\binom{n+i-1}{i}\binom{i}{k}\binom{k}{p}\binom{p}{q}\binom{q}{s}\frac{(3)^i(\frac{-15}{6})^k(\frac{-16}{15})^p(\frac{-1}{2})^q(\frac{-3}{16})^s}{(1-z)^{4i}}z^{i+k+p+q+s}.\end{aligned}$$

The result then follows from developing of $(1-z)^{-4i}$ in series as $(1-z)^{-4i} = \sum_{j\geq 0}\binom{4i+j-1}{j}z^j$ and using the Lagrange inversion formula. □

8.3.2. *Rooted level-2 networks.*

Proposition 8.2. *For any $n \geq 1$, the number ℓ_n of rooted level-2 phylogenetic networks with n leaves is given by*

$$\ell_n = (n-1)! \sum_{\substack{0\leq t\leq m\leq s\leq q\leq p\leq k\leq i\leq n-1\\ j=n-1-i-k-p-q-s-m-t\geq 0\\ i\neq 0}} \begin{array}{l}\binom{n+i-1}{i}\binom{6i+j-1}{j}\binom{i}{k}\binom{k}{p}\binom{p}{q}\binom{q}{s}\binom{s}{m}\binom{m}{t}\\ \times(9)^i\left(\frac{-17}{6}\right)^k\left(\frac{-53}{34}\right)^p\left(\frac{-148}{159}\right)^q\left(\frac{-81}{148}\right)^s\left(\frac{-8}{27}\right)^m\left(\frac{-1}{8}\right)^t.\end{array}$$

Sketch. This follows again from the Lagrange inversion formula, using the equation $L(z) = z\phi(L(z))$ for the function ϕ given in Theorem 7.1. The computations involve the usual development of $(1-z)^{-n}$ given by Eq. (2) and the binomial formula, applied following exactly the same steps as in the proof of Proposition 8.1. Details of the computations are left to the reader. □

(MB) Institut für Mathematik, Universität Zürich, Winterthurerstr. 190, CH-8057 Zrich, Switzerland

E-mail address: mathilde.bouvel@math.uzh.ch

(PG) Université Paris-Est, LIGM (UMR 8049), UPEM, CNRS, ESIEE, ENPC, F-77454, Marne-la-Vallée, France

E-mail address: philippe.gambette@u-pem.fr

(MM) Technische Universität Wien, Department of Discrete Mathematics and Geometry, Wiedner Hauptstrasse 8-10/104, A-1040 Wien, Austria.

E-mail address: marefatollah.mansouri@tuwien.ac.at

Extracting Event-related Information from a Corpus Regarding Soil Industrial Pollution

Chuanming Dong, Philippe Gambette, Catherine Dominguès

Extracting Event-related Information from a Corpus Regarding Soil Industrial Pollution

Chuanming Dong[1,3][a], Philippe Gambette[2][b] and Catherine Dominguès[1][c]
[1]*LASTIG, Univ. Gustave Eiffel, ENSG, IGN, F-77420 Champs-sur-Marne, France*
[2]*LIGM, Univ. Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée, France*
[3]*ADEME, Agence de l'Environnement et de la Maîtrise de l'Énergie, F-49004, Angers, France*
chuanming.dong@ign.fr, philippe.gambette@univ-eiffel.fr, catherine.domingues@ign.fr

Keywords: Information Extraction, Deep Learning, Word Embedding, Semantic Annotation, Industrial Pollution.

Abstract: We study the extraction and reorganization of event-related information in texts regarding industrial pollution. The object is to build a memory of polluted sites that gathers the information about industrial events from various databases and corpora. An industrial event is described through several features as the event trigger, the industrial activity, the institution, the pollutant, etc. In order to efficiently collect information from a large corpus, it is necessary to automatize the information extraction process. To this end, we manually annotated a part of a corpus about soil industrial pollution, then we used it to train information extraction models with deep learning methods. The models we trained achieve 0.76 F-score on event feature extraction. We intend to improve the models and then use them on other text resources to enrich the polluted sites memory with extracted information about industrial events.

1 INTRODUCTION

Pollution is becoming one of the major concerns for French dwellers. The French Ministry of the Ecological Transition (MTES) is responsible for collecting and updating pollution data from industrial sites which are gathered in a certain number of databases, including BASOL, the database of (potentially) polluted sites; BASIAS, a historical inventory of old industrial sites; and S3IC, the database of classified facilities.

With abundant information about industrial sites, these databases are proven to be necessary for the assessment of the situation of a polluted site and the calculation of the cost for rehabilitating a wasteland. Nevertheless, the information contained in them can become inconsistent across databases due to their specific objectives and different update rates. The BASIAS database has been created to record the activities of old industrial sites. Comparing to other databases, it specializes at classifying the productive activities of a site, but in the meantime some information, for example the address of a site, may not be up to date in this database. The S3IC database has been constructed through inspection of industrial facilities, which means it contains the information about the operations of facilities on a site, the authorization status for the operations and the danger level of those facilities. It classifies the industrial activities conducted through those facilities from the point of view of an MTES inspector, which makes S3IC different from other databases about polluted sites. Lastly, BASOL focuses on the pollution of industrial sites. In this database, each site is described in details through the potential pollution processes and/or the remediation processes, as well as a list of pollutants detected in the site, all of these are missing from the other databases.

The multiplication of databases and their content variations make it difficult to have a synthetic view of the situation of the sites. In addition, historical information such as industrial events also plays an important role in the assessment of sites, but this information is either missing or disorganized in these databases.

Therefore, we have planned to create a memory of sites that reorganizes the information from these databases in a more invariable and efficient way. A memory of sites is a database constructed on events that happened in those sites. Eventually, users will be able to query this database for polluted site information, like location, pollutants and industrial activi-

[a] https://orcid.org/0000-0003-3232-8177
[b] https://orcid.org/0000-0001-7062-0262
[c] https://orcid.org/0000-0002-0362-6805

ties etc. Since the existing databases do not share the same objectives regarding the pollution treatment nor the same definition of an industrial event, they do not record the same events. Besides, those events are usually embedded in narrative texts as a part of databases, and there are a lot more events described in the texts, like regulatory reports, rather than in databases. So, in this paper, we introduce an information extraction model which enables event-related extraction from a plain text. In the future, the chronological assembly of these events will make it possible to build the memory of polluted sites.

The information extraction model suits the BASOL narrative texts from which events must be extracted. So, after a brief introduction about related work in section 2, section 3 describes the narrative text corpus, the notion of event and the features which describe industrial events and are looked for in the corpus. The automatic annotation process is based on deep learning; it combines a neural network and word embeddings; they are explained in section 4. The automatic annotation of the event features is assessed in section 5. The results are discussed, based on precision, recall and F-score measures in section 6. The paper concludes with perspectives in section 7.

2 RELATED WORK

In natural language processing (NLP), an information extraction task can be regarded as a sequence labeling task or a classification task. Information extraction tasks focused on event features are relatively new to the NLP community. Over the last decade, several approaches have been proposed by different researchers. In (Arnulphy, 2012), a machine learning model has been used to classify the words by their predefined syntactic, morphologic, semantic and lexical features in order to recognize the events. This classifier is based on a decision tree algorithm, and eventually gets a 0.74 F-score on linguistic feature classification. In (Battistelli et al., 2013), a data mining approach has been proposed, which involves extracting semantic patterns of sentences that describe an event. So, sentences with similar patterns can be extracted as events. Although these approaches are different in usage of models and algorithms, they all require the assistance of an abundant linguistic resource. For example, in (Arnulphy, 2012), French lexicons including action verbs and event nouns are used to define the lexical features of words. In recent years, the development of artificial neural network and language models has made the deep learning approaches much more viable for NLP tasks, including sequence labeling tasks. In (Panchendrarajan and Amaresan, 2018), a model trained on Bi-LSTM neural network has gained a 0.90 F-score on named entity annotation. In the work of (Shin et al., 2020), a spatial information extraction model based on BERT (Bidirectional Encoder Representations from Transformers) is presented. By implementing the language model BERT, the authors have successfully extracted different types of spatial entities with a F-score of 0.90 in total. From these works, it can be seen that the usage of artificial neural networks and language models has improved the result in sequential labeling tasks, especially semantic annotation, without implementing extra linguistic resources.

Our project to build a memory of polluted sites focuses on extracting information about industrial events. As named entity extraction, event extraction is also a semantic annotation task. Different from previous work, we seek to extract events with a certain theme: pollution. This means that we need an approach with strong ability to process semantic features in text. Our proposed approach is inspired by recent work and is based on a deep learning method and a language model.

3 THE BASOL CORPUS AND THE INDUSTRIAL EVENTS

BASOL describes polluted or potentially polluted sites, and soils requiring preventive or remedial action by public authorities through a structured database of the industrial events, which is complemented by narrative texts. The description of industrial event includes specific features, which are relevant in the context of pollution. The corpus extracted from the BASOL database is first presented. The concept of industrial event with its characteristics is based on this corpus; the design of the labels of the characteristics and their use are then introduced.

3.1 Description of the Corpus

BASOL contains structured information about more than 7 000 polluted sites since the 1990s, including their geographic location, owners' identity and detected pollutants. In addition, narrative texts are added to the database records and provide detailed information concerning the facilities and the industrial sites. The texts collected as a corpus provide the source in which industrial events are looked for. The corpus contains 155 587 sentences, with a vocabulary of 48 032 words. The descriptive texts are meant to clarify the industrial incidents that had an influence on

the site, so they include mentions of industrial events. The vocabulary is focused on the topic of the industrial pollution. Since this is an official database, the usage of standard French is also a significant quality. As an example, the following sentence is taken from the corpus: *La société BRODARD GRAPHIQUE était installée depuis 1959 sur la zone industrielle de Coulommiers* (BRODARD GRAPHIQUE was established since 1959 on the Coulommiers industrial area).

3.2 The Concept of Event

The corpus details industrial events. But what exactly is an event? By the definition of dictionary, an event is "*a thing that happens, especially something important*"[1]. Various definitions of an event have been made in previous works. In her doctoral thesis, (Arnulphy, 2012) defines an event as something happens that changes the state. In (Lecolle, 2009), an event is regarded as a singular, unexpected and unrepeatable case. In (Battistelli et al., 2013), although there is no clear definition of *event*, the importance of date in event extraction is emphasized which implies that *event* is a notion with significant temporal properties. From these definitions, it is shown that *event* is a relatively subjective notion which can be adapted to the need of research. But there is a consistency in these definitions. It is clear that the notions of "important" and "happen" are crucial. These notions represent two major aspects of an event: occurrence and importance. From a semantic perspective, occurrence can be interpreted as having a distinctive and closed time range. And importance implies an impact on the reality. Therefore an event can be defined as something that impacts the reality, with a distinctive and terminated time marker.

In this project, we specifically study industrial events. Based on the definition of event, an industrial event can be defined as something impacts the industrial situation, with a distinctive and terminated time marker. According to this definition, several elements must be defined to specify an industrial event. First, to describe the occurrence of an event, a time marker, an action and an actor are required. Since eventually the events will be linked to industrial sites in the database, a place marker is also crucial. With these elements extracted, we can describe the occurrence of an event as "Who did What When and Where". Second, the importance of the event needs to be described. Although the impact on industry can not be extracted directly from a text, information may be found on the influence of an industrial event on the environment. To gather this information, elements such as pollutants, chemical components and products should also be extracted.

[1]https://www.oxfordlearnersdictionaries.com/definition/american_english/event

3.3 Label Design and Application

Therefore, we propose the following set of labels to designate the features of an industrial event:

- **O**: an object, a nominal phrase that serves as an argument of an action. It can be either the actor, the receiver or the complement of an action;
- **N**: an action trigger of an event, usually a momentary verb or its nominal derivation;
- **A**: an industrial activity; An activity is a repeating action that a company conducts daily;
- **T**: an indicator of time, typically a date;
- **L**: an indicator of location, only geographic and administrative locations;
- **R**: a relation, usually a prepositional phrase indicating the logical relation between other labels;
- **I**: an institution's name;
- **S**: a chemical element;
- **U**: a pollutant other than chemical elements;
- **D**: a pollutant in form of a container for other pollutants, for example a wasteyard.

These labels, while covering the need for annotating basic information, may cause a problem of overlap. For example, in this segment that describes an industrial activity, *aspersion de Xylophène sur les poutres de bois* (in English: Xylophene sprinkling on the wooden beams), label **U** should be assigned to the chemical product *Xylophène* (Xylophene), while another label **A**, industrial activity, is assigned to the whole segment. In order to reduce the risk of overlapping, the labels have been separated into 2 groups. The first one contains the labels **O**, **N**, **T**, **A**, **L** and **R**, which are useful to describe an event or an activity. The **I**, **D**, **S** and **U** labels are in the second group; they provide complementary information about pollution and institution. From a linguistic perspective, the labels of the first group have a strong link to syntactic features of words. The assignment of the first group labels requires information about the part-of-speech and the dependency relations between words, such as whether the word is a noun or a verb, whether it is the subject or the predicate in the sentence. The second group is more related to semantic features, and it is by knowing the meaning of the words that these labels can be assigned. For example, *Hydrocarbure* (Hydrocarbon) is identified as a chemical substance (label **S**)

not because it is the subject of a sentence, but because it means *an organic compound consisting entirely of hydrogen and carbon*[2]. In addition, a priority rule has been defined in order to assign only one label to each word. For example, a place name, annotated as a location, L label (first group), may also be annotated **O** (second group) as the object of an event trigger verb. The rule which has been implemented priorizes the indicator of location, which much more specifies the event than the fact it is an object too.

On the other hand, the designation of the event features are often made up of several words, for example: *La société BRODARD GRAPHIQUE*, *sur la zone industrielle de Coulommiers*. Therefore, the "B-I-E-O" (begin, inside, end, outside) annotation format has been implemented in the annotation work. Since this format uses different labels for the beginning and the end of an extracted expression, it enables to detect multiword units. In this way, both category labels and boundary labels can be assigned at the same time to each word in a group. So, it is easy to distinguish between groups of words, even if they are of the same category. Consequently, the labels assigned to each word is in fact a combination of a boundary label and a category label. Here is an example:

Les	*installations*	*de*	*l'usine*
BO	**IO**	**IO**	**EO**

ont	*été*	*démolies*	*entre*	*1970*	*et*	*1980* .
BN	**IN**	**EN**	**BT**	**IT**	**IT**	**ET**

The two-character labels enable to delimit three phrases: *Les installations de l'usine* (label **O**), *ont été démolies* (label **N**), and *entre 1970 et 1980* (label **T**).

As can be seen in this example, the assignment of labels is realised within a sentence. Normally, the boundary of a sentence does not necessarily match the boundary of an event; some features of an event may appear in a different sentence from the one that contains the trigger of the event. However, the BASOL corpus is a combination of brief texts that summarize the activities and events that occur at a site. So, it is more unlikely to find an event announced in two sentences in this corpus. Consequently, the narrative texts have been segmented and annotated into sentences. This has several advantages. The sentence is a perfect unit for the input of a deep learning algorithm (see the next section), since a paragraph as a unit may be too voluminous for the algorithm to run efficiently, and a word as a unit risks loosing context features of the word. The segmentation into sentences enables a better control of the manual annotation workload.

[2]https://en.wikipedia.org/wiki/Hydrocarbon

4 AUTOMATIC ANNOTATION OF EVENT FEATURES

The targeted memory of polluted sites is based on a chronological assembly of pollution events. Each of them is described through its features; the goal of the information extraction model is to automatically identify and annotate the features. The model which is proposed combines a neural network to identify the phases, and word embeddings to distinguish between the use contexts of each word occurrence. The two components are independent and the choice of each one is guided by criteria that are explained. The training of the model combines both components and is based on the training corpus that has been manually annotated.

4.1 Choice of the Information Extraction Model

Several models are suitable to automatic information extraction. The most adopted ones are the models based on linguistic rules, and those trained with supervised deep learning method.

The rule-based models can perform a very precise information extraction. However, they rely on implemented vocabularies and their performance may deteriorate when processing a corpus with new terminologies, which is known as an Out-of-Vocabulary problem (OOV). This could be a major drawback in our case because the corpus could be extended to other documents that deal with the same theme but with another vocabulary (more technical or more regulatory) or with the mention of new institution names and other chemical product names. Finally, we choose to make a neural model based on deep learning method, in order to solve OOV and to obtain a more flexible tool. The supervised deep learning method on which the model is made is called Bi-LSTM (Bidirectional Long Short-Term Memory) (Basaldella et al., 2018). LSTM is a recurrent neural network (RNN). Comparing to other neural network structures, RNN is more suitable for sequential learning task, especially in the case where the output of an input can be influenced by the previous inputs. This property of RNN suits the feature annotation since a word's label assignment is strongly influenced by the words in its context. Derived from the traditional RNN, the Bi-LSTM neural network is more flexible than RNN in sequence tagging tasks because of its ability of reserving the influ-

ence of a word's remote context during training. And since this is a bi-directional model, it can learn from both previous context and following context, and thus it is more suitable for detecting the beginning and the end boundaries of an expression.

4.2 Choice of Word Embeddings

For text data being able to be processed by the neural network, one step is indispensable: word embedding. Indeed, every input text word is substituted with its vector that the algorithm can process. So, the vector returns the context of the word in the text. Several word integration models exist, which influence the performance of the information extraction models. At the beginning of the implementation, in order to quickly test the performance of Bi-LSTM neural network, we have tried training with one of the simplest word embedding method: Word2vec (Mikolov et al., 2013). This method, while able to efficiently provides word vectors generated from the context of each word, has some fundamental flaws that influenced the performance of the models. First of all, the vector generated by Word2vec is static, this means each word form has one and only one vector for the whole text unit, regardless of its different contexts. Consequently, the word vectors generated by Word2vec model cannot represent polysemy, the case where a word can have different meanings in different context. Furthermore, unlike multi-layer deep learning word embedding models, Word2vec cannot generate vectors that embed complex linguistic information of different levels, such as a word's syntactic and semantic features. Therefore, other word embedding models have been taken into consideration, specially some state of art language models. Finally, we have decided to use the French language model CamemBERT (Martin et al., 2020), a Transformer-based model trained on a large French corpus. This model is known for its state-of-art performance for natural language processing tasks in French, including part-of-speech tagging, dependency parsing and named entity recognition. What makes this model special is that it assigns different vectors to different occurrences of the same word, according to the contexts. And for words it cannot recognize, it breaks down the words into morphemes to assign them the corresponding vectors. Thus, this model is not affected by polysemy or OOV problems. Since this model can efficiently integrate the semantic features in the context, it would be helpful for recognizing the labels closely related to word sense, the pollutants for example.

4.3 Training and Validation Corpora

As explained above, the proposed model is based on a neural bi-LSTM model. It must be trained with an annotated corpus in order to learn the labels which annotate the event features. The annotated corpus must be reliable (annotations must be manually checked), consistent, suitable for the task and of sufficient size. In addition, a part of the manually annotated corpus must be reserved for the assessment task. In order to reduce the manual annotation work, a "bootstrapping" annotation-training process has been implemented. First, the event-related information is manually annotated in a small sample of corpus. Then, the model is trained on this annotated sample to become a rough trained annotation model. Through this model, another corpus sample can be automatically annotated and then manually corrected, resulting in a new training cycle for the model, which improves it. By repeating this process we can perform a "bootstrapping" annotation-training process. It enables to accumulate annotated and checked samples which are gathered to form the final training corpus. Thus, the model can be trained, as much as necessary, on an abundant and reliable corpus and become an efficient tool.

As seen before, the narrative texts have been segmented into sentences and annotated. Thus, each input data unit of the model is a sentence which is in the form of a tensor that contains the vector of every sentence word.

The passage from a sentence to its words is based on a tokenization process. To ensure the coherent combination of the different components of the final model, the tokenization method of the word embedding provider, i.e. CamemBERT, has been adopted. However, the way that CamemBERT splits certain words into lexemes can cause inconvenience for manual annotation or correction. Therefore, a script that can transform the CamemBERT tokens to TreeTagger (Schmid, 1994) tokens[3] has been prepared, along with their labels. The TreeTagger tokenization is the one chosen for the manual annotation, but this script can also transform CamemBERT tokens to any other types of tokens. The script can also work in the opposite direction, and transform other tokenized sentences to CamemBERT tokens.

This is a bootstrapping experiment that augments the annotated text through the model training sessions. For the first session, only 120 annotated sentences were prepared for training the model, and 100 sentences to test and evaluate it. After applying the model, we manually corrected the annotation result,

[3] https://github.com/DongChuanming/KDIR_2021_shared/blob/main/KDIR_tokenization_transformer.py

and thus obtained 100 more correctly annotated sentences.

The second session has consisted of several steps: first, a transitory model has been trained on the 220 annotated sentences, then by using this model, 301 new sentences have been automatically annotated. This enables to efficiently obtain 301 more parsed sentences by correcting the annotation result. Then these sentences have been split into 3 groups: 130 sentences join the training data, giving 350 sentences for model training; 120 sentences for developing, more precisely for choosing the number of epochs; and the evaluation set composed of those 120 sentences complemented with the last 51 annotated sentences.

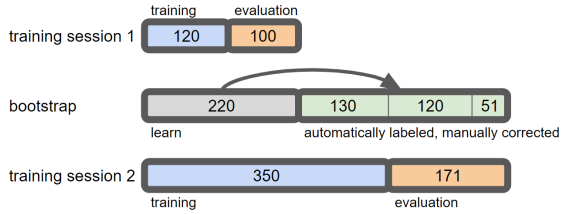


Figure 1: Illustration of the bootstrap method used to augment the training and evaluation corpora.

5 EVALUATION OF ANNOTATION MODELS

Since the labels have been separated into two groups, two models (named Model 1 and Model 2) have been implemented to automatically annotate the event features. Both are based on the Bi-LSTM neural algorithm and use the same word embeddings provided by CamemBERT. They have been trained and assessed with the same training and evaluation corpora. They share the same training processes (numbers of epochs and learning rate), named session below.

During model training, the evaluation has already begun. In order to find the parameters that optimize the training, we have tested the models with 120 developing sentences with different network configurations. To illustrate, here is a graph that shows how the F-score of each label of Model 1 evolves according to different numbers of epochs, with learning rate at 0.01 :

According to figure 1, at epoch 400, most labels have the highest F-score, thus 400 is the best epoch number for Model 1 training if other parameters don't change. Aside from epoch number, we have also tested other parameters like learning rate and butch size, for both Model 1 and Model 2, to find their best value. The evaluation results presented below are for

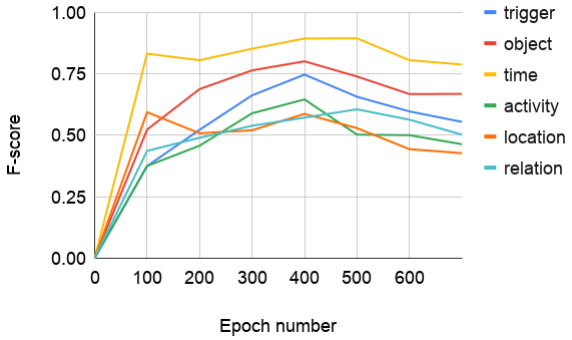


Figure 2: Evolution of the F-score computed on the developing set of the first session, by epoch number for each label - Model 1.

models trained with the best parameters at the moment. Since all parameters have not yet been tested, it is possible that the models will be further improved. The evaluation results of the two models trained during both sessions are shown in the following tables. The evaluation is realised on each label separately. Since event-related information has been extracted through the category labels, at this stage, the boundaries labels have not been evaluated. Table 1 and 2 are the evaluation of Model 1 and Model 2 trained during the first session.

Table 1: Number of true positives (TP) and evaluation of the precision (p), recall (r) and F-score of Model 1 on the test set of the first training session (100 sentences, 400 epochs).

Label	TP	p	r	F-score
trigger (**N**)	143	0.66	0.57	0.61
activity (**A**)	64	0.40	0.58	0.47
object (**O**)	209	0.94	0.85	0.89
time (**T**)	184	0.93	0.88	0.90
location (**L**)	61	0.63	0.59	0.61
relation (**R**)	23	0.45	0.45	0.45
Total	684	0.72	0.70	0.71

Table 2: Result of Model 2 on the test set of the first training session (100 sentences, 400 epochs).

Label	TP	p	r	F-score
institution (**I**)	28	0.93	0.46	0.62
chemicals (**S**)	29	0.88	0.58	0.70
pollutant (**P**)	2	0.10	0.40	0.16
container (**D**)	0	-	-	-
Total	59	0.71	0.50	0.59

Table 3 and 4 show the evaluation of Model 1 and Model 2 trained during the second session.

Table 3: Result of Model 1 on the evaluation set of the second training session (171 sentences, 400 epochs).

Label	TP	p	r	F-score
trigger (**N**)	308	0.77	0.69	0.73
activity (**A**)	120	0.62	0.64	0.63
object (**O**)	763	0.89	0.82	0.85
time (**T**)	194	0.89	0.93	0.91
location (**L**)	86	0.55	0.63	0.59
relation (**R**)	157	0.61	0.54	0.57
Total	1628	0.78	0.74	0.76

Table 4: Result of Model 2 on the evaluation set of the second training session (171 sentences, 400 epochs).

Label	TP	p	r	F-score
institution (**I**)	146	0.95	0.77	0.85
chemicals (**S**)	95	0.90	0.82	0.86
pollutant (**P**)	54	0.75	0.47	0.57
container (**D**)	2	0.25	0.15	0.19
Total	297	0.87	0.68	0.77

6 RESULT ANALYSIS

Although we only used a small manually annotated corpus, we already obtained promising results on the models. For a simple comparison, we have also tested two other NLP tools on date annotation, a popular Python library called `dateparser`[4], and `NOOJ`[5], an annotation software for linguists. Both of them are based on rules. Considering the reliance of event on its time marker, this comparison should be able to reflect the performance on event extraction too. As a result, `dateparser` can only detect the date expressions in our text with a 0.77 precision and a 0.48 recall; `NOOJ` obtained 0.98 precision, but only a 0.44 recall. This proves that our models have a state-of-art performance for detecting certain entities. By observing the score of the different labels, and by comparing the manual and automatic annotations, we have discovered some interesting points to address. The score of the different labels, and the comparison between the manual and automatic annotations give clues to improve the results of the automatic annotation of the event features. The commentaries of the results and the improvement clues are organized regarding three themes : the confusion between labels, the improvement due to the increase of the corpus, and the relevance of the CamemBERT word embeddings.

[4]https://dateparser.readthedocs.io/en/latest/

[5]http://explorationdecorpus.corpusecrits.huma-num.fr/nooj/

6.1 Comparison between Labels

The models do not work well on some labels. Comparing to time (**T**) and object (**O**) labels, event trigger (**N**), industrial activity (**A**) and location (**L**) labels do not have an impressive F-score. After observing the automatic annotation results on these labels, we see that certain sentences that should have been annotated as event trigger, are annotated as industrial activity. Based on our definition of event trigger, the action that triggers an event should be a momentary verb or its nominal derivation. In contrast, an industrial activity is an action conducted by enterprises frequently during a period of time, and should be designated by a durative verb or its nominal derivation, or a repeating action. However, it is difficult to distinguish an event trigger expression from an industrial activity expression, based on their syntactic features, especially when they are all nominal derivation of verbs. Unlike a verb, a noun does not have "momentary" nor "durative" as properties. Therefore, once nominalized, these event trigger expressions are confused with an activity, usually in the form of a nominal phrase.

A similar problem can be found with the label *location*. Since the expression of a location often has a prepositional structure, the nominal part of a location expression can easily be recognized as an *object* if its position is close to an event trigger or an activity. Besides, based on its definition, the recognition of a location expression is trickier. The location expressions we want to extract include only geographic and administrative locations. For example, even if the prepositional phrase *dans les nappes des calcaires grossiers* (in the coarse limestone sheets) indicates a position and hence is annotated by our model as a *location*, it does not belong to either precedent types , and therefore should not be recognized as a location.

6.2 Improvement Due to the Corpus Increase

An improvement can be observed between the two sessions. Comparing to the first session, the models trained in second session have a better performance on annotating most labels due to the increase of the training text. Also, it is noticeable that Model 2 has benefited more from this training corpus increase. Indeed, the labels of the second group are less frequent than those of the first group. Consequently, there are not enough second group annotation examples in the first session; the category *container* (**D**) is even absent from the test corpus of the first session. With more training text attached to the second session, the models are able to learn the second group annotations

on more label instances and thus improve Model 2.

6.3 Relevance of the Word Embeddings

The use of the CamemBERT word embeddingds also improved the results. The *pollutant* category (**P**) is the one that benefits the most from the use of the vectors. To compare, by using our preliminary model implementing the Word2vec method, the pollutant annotation precision is only 0.05 but by using the current model the score has increased to 0.56 without lowering the recall. A pollutant expression is usually a nominal phrase. It is very difficult to differ it from any other nominal component, on syntactic level. And unlike institution names or chemicals, the expression of pollutants does not involve changes of word case or the usage of nomenclatures. So the most promising ways to recognize them are by analysing the polarity (positive or negative) in the context, and by building the word meaning itself, all of which require usage of complicated semantic features. Unlike syntactic features, semantic features are hard to extract and to be comprehended by the algorithm. The CamemBERT model, which has embedded semantic features in form of word vectors, enables the neural network to learn annotation patterns on a semantic level. So, our model can recognize some typical pollutant expressions, like *tensio actif* (surfactant) and other chemical products, which is exactly the information which must be extracted in order to build the memory of polluted sites.

7 CONCLUSION

In this paper, we have described an approach for event-related information extraction from a corpus focused on industrial pollution. With a supervised deep learning method, we trained two models that can simulate our manual annotation on industrial event features. Right now, the models trained with Bi-LSTM neural networks have given promising results, but we still need them to be better at detecting event triggers and industrial activities in order to use them on other text resources. Given the fact that the models are trained with only a small portion of the corpus, and the neural network configurations are not fully explored, it could be possible to improve the model. Aside from increasing training text data and adjusting neural network setting, it is also interesting to see if the model could have a better performance if we use paragraphs instead of sentences as the input of the neural networks, since the narration of an event is not limited in a sentence.

This work is devoted to the construction of the polluted sites memory, based on an only consistent and complete database. Eventually, the event-related information extracted by the models will be inserted in the database. For future work, we will apply a syntactic parser to link the extracted event features by dependency relations, and train a classifier to categorize the events, so that they can be integrated into the database with an appropriate structure. The models will also be tested and used on other corpora in the domain of industrial pollution, to connect other sources of data and enrich the polluted site memory.

REFERENCES

Arnulphy, B. (2012). *Désignations nominales des événements: étude et extraction automatique dans les textes*. PhD thesis, Université Paris 11.

Basaldella, M., Antolli, E., Serra, G., and Tasso, C. (2018). *Bidirectional LSTM Recurrent Neural Network for Keyphrase Extraction*, pages 180–187. Springer.

Battistelli, D., Charnois, T., Minel, J.-L., and Teissèdre, C. (2013). Detecting salient events in large corpora by a combination of NLP and data mining techniques. In *Conference on Intelligent Text Processing and Computational Linguistics*, volume 17(2), pages 229–237, Samos, Greece.

Lecolle, M. (2009). Éléments pour la caractérisation des toponymes en emploi événementiel. In Evrard, I., Pierrard, M., Rosier, L., and Raemdonck, D. V., editors, *Les sens en marge Représentations linguistiques et observables discursifs*, pages 29–43. L'Harmattan.

Martin, L., Muller, B., Ortiz Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É., Seddah, D., and Sagot, B. (2020). CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Panchendrarajan, R. and Amaresan, A. (2018). Bidirectional LSTM-CRF for named entity recognition. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees.

Shin, H. J., Park, J. Y., Yuk, D. B., and Lee, J. S. (2020). BERT-based spatial information extraction. In *Proceedings of the Third International Workshop on Spatial Language Understanding*, pages 10–17, Online. Association for Computational Linguistics.

Reordering a tree according to an order on its leaves

Laurent Bulteau, Philippe Gambette, Olga Seminck

Reordering a tree according to an order on its leaves

Laurent Bulteau
LIGM, Université Gustave Eiffel & CNRS, Champs-sur-Marne, France

Philippe Gambette[1]
LIGM, Université Gustave Eiffel & CNRS, Champs-sur-Marne, France

Olga Seminck
Lattice (Langues, Textes, Traitements informatiques, Cognition), CNRS & ENS/PSL & Université Sorbonne nouvelle, France

Abstract

In this article, we study two problems consisting in reordering a tree to fit with an order on its leaves provided as input, which were earlier introduced in the context of phylogenetic tree comparison for bioinformatics, OTCM and OTDE. The first problem consists in finding an order which minimizes the number of inversions with an input order on the leaves, while the second one consists in removing the minimum number of leaves from the tree to make it consistent with the input order on the remaining leaves. We show that both problems are NP-complete when the maximum degree is not bounded, as well as a problem on tree alignment, answering two questions opened in 2010 by Henning Fernau, Michael Kaufmann and Mathias Poths. We provide a polynomial-time algorithm for OTDE in the case where the maximum degree is bounded by a constant and an FPT algorithm in a parameter lower than the number of leaves to delete. Our results have practical interest not only for bioinformatics but also for digital humanities to evaluate, for example, the consistency of the dendrogram obtained from a hierarchical clustering algorithm with a chronological ordering of its leaves. We explore the possibilities of practical use of our results both on trees obtained by clustering the literary works of French authors and on simulated data, using implementations of our algorithms in Python.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases tree, clustering, order, permutation, inversions, FPT algorithm, NP-hardness, tree drawing, OTCM, OTDE, TTDE

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material `https://github.com/oseminck/tree_order_evaluation`

Funding *Philippe Gambette*: "Investissements d'avenir" program, reference ANR-16-IDEX-0003 (I-Site Future, programme "Cité des dames, créatrices dans la cité").
Olga Seminck: "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute)

[1] corresponding author

1 Introduction

The problem of optimizing the consistency between a tree and a given order on its leaves was first introduced in bioinformatics in the context of visualization of multiple phylogenetic trees in order to highlight common patterns in their subtree structure [6], under the name "one-layer STOP (stratified tree ordering problem)". The authors provided an $O(n^2)$ time algorithm to minimize, by exchanging the left and right children of internal nodes, the number of inversions between the left-to-right order of the leaves of a binary tree and an input order on its leaves. The problem was renamed OTCM (One-Tree Crossing Minimization) in [9], where an $O(n \log^2 n)$ time algorithm is provided, as well as a reduction to 3-Hitting Set of a variant of the problem where the goal is to minimize the number of leaves to delete from the tree in order to be able to perfectly match the input order on the remaining leaves, called OTDE (One-Tree Drawing by Deleting Edges). An $O(n \log^2 n / \log \log n)$ time algorithm is later provided for OTCM by [1], improved independently in 2010 by [10] and [22] to obtain an $O(n \log n)$ time complexity. About OTDE, the authors of [10] note that "the efficient dynamic-programming algorithm derived for the related problem OTCM [...] cannot be transferred to this problem. However, we have no proof for NP-hardness for OTDE nor TTDE, either". TTDE (Two-Tree Drawing by Deleting Edges) is a variant of OTDE where two leaf-labeled trees are provided as input and the goal is to delete the minimum number of leaves such that the remaining leaves of both trees can be ordered with the same order. We give below an answer to both sentences, providing a dynamic-programming algorithm solving OTDE for trees with fixed maximum degree as well as an NP-hardness proof in the general case for OTDE and for TTDE.

Although this problem was initially introduced in the context of comparing tree embeddings, one tree having its embedding (that is the left-to-right order of all children) fixed, we can note that only the order on the leaves of the tree with fixed embedding is useful to define both problems OTCM and OTDE. Both problems therefore consist not really in comparing trees but rather in reordering the internal nodes of one tree in order to optimize its consistency with an order on its leaves provided as input. A popular problem consisting in finding an optimal order on the leaves of a tree is "seriation", often used for visualization purposes [7], where the optimized criterion is computed on data used to build the tree. For example, a classical criterion, called "optimal leaf ordering", is to maximize the similarity between consecutive elements in the optimal order [2, 3, 4]. Another possibility is to minimize a distance criterion, the "bilateral symmetric distance", computed on pairs of elements in consecutive clusters [5]. Seriation algorithms have been implemented for example in the R-packages `seriation` [12] and `dendsort` [19].

With the OTCM and OTDE problems, our goal is not to reorder a tree using only the original data from which it has been built, but using external data about some expected order on its leaves. In the context where the leaves of the tree can be ordered chronologically, for example, this would help providing an answer to the question: how much is this tree consistent with the chronological order? This issue is relevant for several fields of digital humanities, when objects associated with a publication date are classified with a hierarchical clustering algorithm, for example literature analysis [14], political discourse analysis [15] or language evolution [17], as noticed in [11]. In these articles, the comments about the chronological signal which can be observed in the tree obtained from the clustering algorithm are often unclear or imprecise. For example, in [17], the author observes about Figure 15 on page 17 that "the cluster tree gives a visual representation consistent with what is independently known of the chronological structure of the corpus". However, the structure of the tree

does not perfectly reflect the chronology[2]. The algorithms solving the OTCM and OTDE problems can also prevent researchers from claiming having obtained perfect chronological trees with clustering, whereas there are still small inconsistencies that are not easy to spot with the naked eye. For example, although "Chez Jacques Chirac, l'examen des parentés [dans ses discours de vœux] ne suppose aucune rupture, la chronologie étant parfaitement représentée"[3] is claimed about Figure 2.4 in [15], the 1999 speech cannot be ordered between 1998 and 2000.

In this article, we first give useful definitions in Section 1.1. We answer two open problems from [10], proving that OTDE and TTDE are NP-complete, as well as OTCM, in Section 2. We then provide a dynamic programming algorithm solving OTDE in polynomial time for trees with fixed maximum degree in Section 3. This algorithm also works in the more general case where the order on the leaves is not strict. We then provide an FPT algorithm for the OTDE problem parameterized by the deletion-degree of the solution, which is lower than the number of leaves to delete, in Section 4. We also give an example of a tree and an order built to have a distinct solution for the OTCM and OTDE problems in Section 5. Finally, we illustrate the relevance of this problem, and of our implementations of algorithms solving them, for applications in digital humanities, with experiments on trees built from literary works, as well as simulated trees, in Section 6.

1.1 Definitions

Given a set X of elements, we define an X-tree T as a rooted tree whose leaves are bijectively labeled by the elements of X. The set of leaves of T is denoted by $L(T)$ and the set of leaves below some vertex v of T is denoted by $L(T, v)$ (or simply $L(v)$ if T is clear from the context). A set of vertices of T is *independent* if no vertex of T is an ancestor of another vertex of T.

We say that σ is a strict order on X if it is a bijection from X to $[1..n]$ and that it is a weak order on X if it is a surjection from X to $[1..m]$, where $|X| \geq m$. Given any (strict or weak) order σ, we denote by $a \leq_\sigma b$ the fact that $\sigma(a) \leq \sigma(b)$ and by $a <_\sigma b$ the fact that $\sigma(a) < \sigma(b)$. Considering the elements $x_1, \ldots, x_n$ of X such that for each $i \in [1..n-1], \sigma(x_i) \leq \sigma(x_{i+1})$, we denote by $(x_1 x_2 \ldots x_n)$ the (weak or strict) order σ.

Given an X-tree T and a (weak or strict) order σ on X, we say that an independent pair $\{u, v\}$ of vertices of T is a *conflict wrt.* σ if there exist leaves $a, c \in L(u)$ and $b \in L(v)$ such that $a <_\sigma b <_\sigma c$. Conversely, if $\{u, v\}$ is not a conflict, then either $a \leq_\sigma b$ for all $a \in L(u), b \in L(v)$, or $b \leq_\sigma a$; we then write $u \preceq_\sigma v$ or $v \preceq_\sigma u$, respectively. We say that σ is *suitable* on T if T has no conflict with respect to σ.

Given two (strict or weak) orders σ_1 and σ_2 on X and two elements $a \neq b$ of X, we say that $\{a, b\}$ is an *inversion* for σ_1 and σ_2 if $a \leq_{\sigma_1} b$ and $b <_{\sigma_2} a$, or $b \leq_{\sigma_1} a$ and $a <_{\sigma_2} b$.

Given an X-tree T, a subset X' of X and an order σ on X, we denote by $\sigma[X']$ the order σ restricted to X', and by $T[X']$ the tree T restricted to X', that is the X'-tree obtained from T by removing leaves labeled by $X \setminus X'$ and contracting any arc to a non-labeled leaf, any arc from an out-degree-1 vertex. We define the *deletion-degree* of X' as the maximum degree of the tree induced by the deleted leaves, i.e., $T[X \setminus X']$. Intuitively, the deletion-degree measures how deletions in different branches converge on a few nodes or if they merge

[2] For example `1380Gawain.txt` cannot be ordered between `1375AllitMorteArthur.txt` and `1400YorksPlays.txt`.

[3] "For Jacques Chirac, the examination of the genealogy [of his new year addresses] shows no discontinuity, the chronology being perfectly represented"

progressively. Note that by definition, the deletion-degree of X' is upper-bounded both by the maximum degree of T and by the size of $X \setminus X'$.

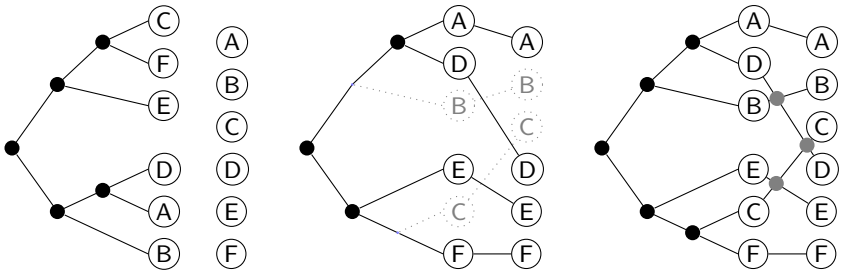


Figure 1 Example for the OTDE and OTCM problem. Left: a tree T on leaves $\{\mathsf{A}, \ldots, \mathsf{F}\}$, the reference permutation is $\sigma = (\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F})$ (more precisely, $\sigma(\mathsf{A}) = 1, \ldots, \sigma(\mathsf{F}) = 6$). Middle: a solution for OTDE with cost 2. The subtree $T[X']$ for $X' = \{\mathsf{A}, \mathsf{D}, \mathsf{E}, \mathsf{F}\}$ is ordered to show the absence of conflicts with $\sigma[X']$. Right: a solution for OTCM with cost 3. The order $\sigma' = (\mathsf{A}, \mathsf{D}, \mathsf{B}, \mathsf{E}, \mathsf{C}, \mathsf{F})$ is suitable for T and yields three inversions with σ.

We now define the two main problems addressed in this paper (see Figure 1 for an illustration). As explained in the introduction, we differ from previous definitions which considered two trees, one with a fixed order on the leaves, as input, as only the leaf order of the second tree is useful to define the problem and not the tree itself.

We therefore define the OTCM (One-Tree Crossing Minimization) problem as follows:

- **Input**: An X-tree T, an order σ on X and an integer k.
- **Output**: Yes if there exists an order σ' on X suitable on T such that the number of inversions for σ' and σ is at most k, no otherwise.

We also define the OTDE (One-Tree Drawing by Deleting Edges) problem as follows:

- **Input**: An X-tree T, an order σ on X and an integer k.
- **Output**: Yes if there exists a subset X' of X of size at least $|X| - k$ such that $\sigma[X']$ is suitable on $T[X']$, no otherwise.

We finally define the TTDE (Two-Tree Drawing by Deleting Edges) problem in the following way:

- **Input**: Two X-trees T_1 and T_2 and an integer k.
- **Output**: Yes if there exists a subset X' of X of size at least $|X| - k$ and an order σ' on X' that is suitable on $T_1[X']$ and on $T_2[X']$, no otherwise.

2 NP-hardness

2.1 OTDE and TTDE are NP-complete for trees with unbounded degree

▶ **Theorem 1.** *The OTDE problem is NP-complete for strict orders and therefore for weak orders.*

Proof. First note that OTDE is in NP, since, given an X-tree T, an order σ and a set L of leaves to remove, we can check in linear time, by a recursive search of the tree, saving on each node the minimum and the maximum leaf in $\sigma[X - L]$ appearing below, whether

$\sigma[X-L]$ is suitable on $T[X-L]$. Regarding NP-hardness, we now give a reduction from Independent Set, which is NP-hard on cubic graphs [16], to OTDE when the input trees have unbounded degree.

We consider an instance of the Independent Set problem, that is a cubic graph $G=(V=\{v_1,\dots,v_n\},E)$ such that $|E|=m=3n/2$ and an integer k. For each vertex v_i, we write e_i^1, e_i^2 and e_i^3 for the three edges incident with v_i (ordered arbitrarily).

We now define an instance of the OTDE problem. The set of leaf labels consists of *vertex labels* denoted v_i and v_i' for each $i\in[1..n]$, one *edge label* for each edge (also denoted e_i^j for the jth edge incident on vertex v_i), and a set of n^2 *separating labels* $B_i=\{b_i^1,b_i^2,\dots b_i^{n^2}\}$ for each $i\in[1..n-1]$.

First, we define the strict order $\sigma(G)=(v_1e_1^1e_1^2e_1^3v_1'b_1^1b_1^2\dots b_1^{n^2}v_2e_2^1e_2^2e_2^3v_2'b_1^2b_2^2\dots b_{n-1}^{n^2}v_ne_n^1e_n^2e_n^3v_n')$. Then, let T_{v_i} be the tree with leaves v_i and v_i' attached below the root, T_e be the tree with leaves $e_i^{i'}$ and $e_j^{j'}$ attached below the root for each edge $e=\{v_i,v_j\}$ of G (with $i',j'\in[1..3]$), and T_{B_i} be the tree with leaves $b_i^1,\dots,b_i^{n^2}$ attached below the root for each $i\in[1..n-1]$. We finally define $T(G)$ as the tree such that $T_{v_1},T_{v_2},\dots,T_{v_n},T_{e_1},T_{e_2},\dots T_{e_m},T_{B_1},T_{B_2},\dots$ and $T_{B_{n-1}}$ are attached below the root.

We claim that G has an independent set of size at least $k\Leftrightarrow$ the instance $(T(G),\sigma(G))$ of the OTDE problem has a solution with a set L of at most $m+n-k$ leaves to remove.

$\Rightarrow$: Suppose that there exists a size-k independent set $S=\{s_1,\dots,s_k\}$ of G. We then remove the following leaves (also contracting along the way the edge from their parent to the root of $T(G)$) in order to get a new tree T':

- for each edge $e=\{v_i,v_j\}=e_i^{i'}=e_j^{j'}$ with $i<j$, we remove $e_i^{i'}$ and call $T_{e_j^{j'}}=T_e$ if $v_i\in S$ or if neither v_i nor v_j belong to S; and we remove $e_j^{j'}$ and call $T_{e_i^{i'}}=T_e$ if $v_j\in S$ (as S is an independent set we cannot have both v_i and v_j in S);
- for each vertex v_i not in S we remove v_i'.

By ordering the children of the root of $T(G)$ such as in Figure 2(1), that is by putting, for each v_i with $i\in[1..n]$, T_{v_i}, then $T_{e_i^1}$, $T_{e_i^2}$ and $T_{e_i^3}$ for each of the $e_i^{i'}$ which were not removed and then T_{B_i} (except for $i=n$), the order $\sigma(G)$ restricted to the remaining $m+n+k+n^2(n-1)$ leaves is suitable on T'.

$\Leftarrow$: Suppose that there exists a set L of at most $m+n-k$ leaves such that $\sigma(G)[X-L]$ is suitable on $T(G)[X-L]$. For each parent p_{B_i} of the leaves of B_i and any other vertex v of T such that $\{p_{B_i},v\}$ is a conflict wrt. $\sigma(G)$, we can delete this conflict either by deleting no leaf of B_i or all leaves of B_i. As each B_i has size $n^2>m+n-k$, its leaves cannot belong to the set L of leaves to be deleted.

We now consider the trees T_{e_i} for each $i\in[1..m]$: by construction of $\sigma(G)$, as both leaves of each such tree are separated by some $B_{i'}$, therefore by $n^2>m+n-k$ leaves, one of these two leaves has to be removed, so it has to belong to L. We call L' the set of such leaves of L, therefore there exists a set $L-L'$ of at most $n-k$ other leaves to delete. So there exists a subset S_L of $[1..n]$ of size at least k such that for any element $i\in S_L$, neither v_i, nor v_i', nor any of the leaves e_i^j for $j\in\{1,2,3\}$ belong to $L-L'$. Note that for such $i\in S_L$, all vertices v_i and v_i' are not in L and all e_i^j are in L'. We claim that the vertices of G corresponding to S_L are an independent set of G. Suppose for contradiction that it is not the case, then there exists an edge $e=e_i^{i'}=e_j^{j'}$ between two vertices v_i and v_j of G. By construction of L', exactly one of the leaves labeled by $e_i^{i'}$ and $e_j^{j'}$ is in L' so the second one is in $L-L'$: contradiction. ◀

▶ **Corollary 2.** *The TTDE problem is NP-complete.*

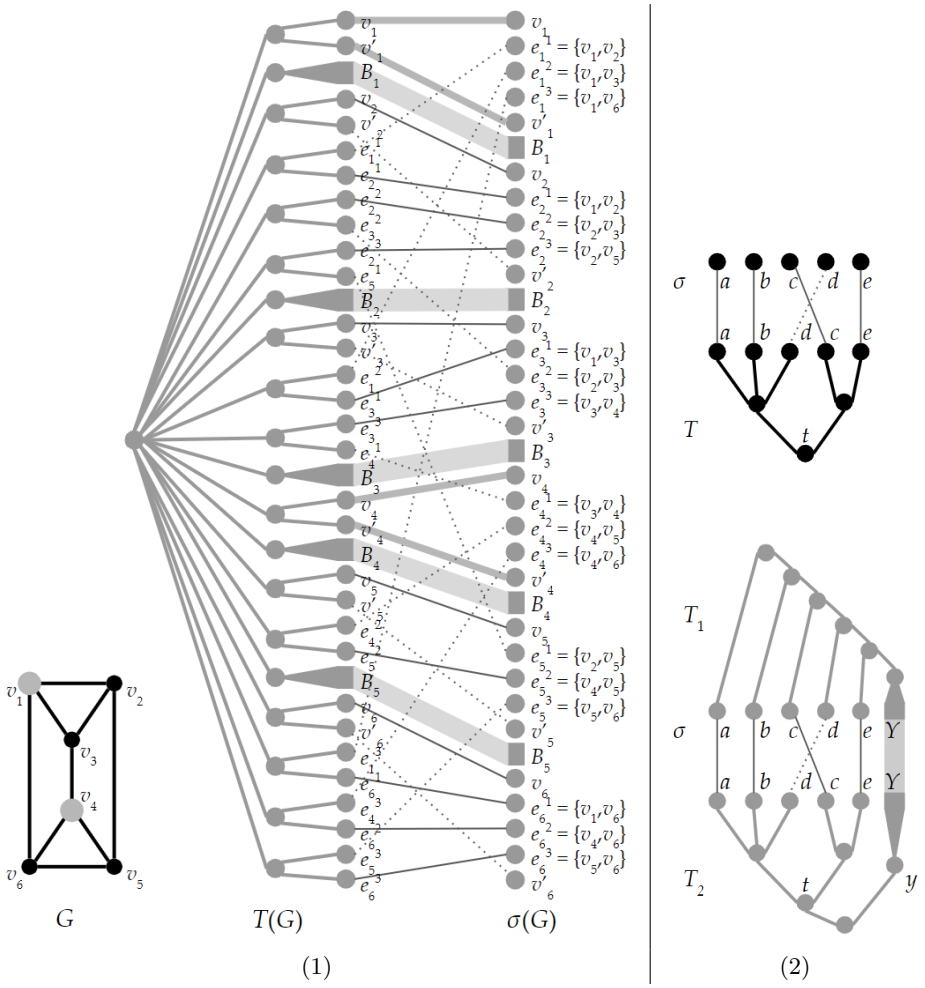


Figure 2 Illustration of the reductions of Independent Set to OTDE and of OTDE to TTDE. (1, left) A graph G with independent set $S = \{v_1, v_4\}$ of size 2. (1, right) The corresponding tree $T(G)$ as well as the order $\sigma(G)$. By removing all leaves connected with dotted lines to the corresponding element in $\sigma(G)$, the resulting subtree of $T(G)$ is suitable for the order (since the remaining arcs are non-crossing). (2) Reduction from an OTDE instance (T, σ) (top) to a TTDE instance (T_1, T_2) (bottom). A large set of leaves labelled Y can be seen as a fixed-point, around which T_1 must be ordered according to σ, and T_2 according to the input tree T.

Proof. TTDE is clearly in NP. We prove hardness by reduction from OTDE (see Figure 2(2) for an illustration). Consider an instance (T, σ) of OTDE with σ a strict order on n labels X. Introduce a set Y of n new labels. Build T_1 as a caterpillar with $n+1$ internal nodes forming a path $r_1, \ldots, r_{n+1}$ (with root r_1) and $2n$ leaves where each r_i with $i \leq n$ has one leaf attached with label $\sigma^{-1}(i) \in X$ (in the same order), and r_{n+1} has n leaves attached labelled with Y. Build T_2 as a tree, where the root has two children y, t, where y has n children which are leaves labelled with Y, and t is the root of a subtree equal to T.

We now show our main claim: given $0 \leq k < n$, OTDE(T, σ) admits a solution with at most k deletions $\Leftrightarrow$ TTDE(T_1, T_2) admits a solution with at most k deletions.

$\Rightarrow$ Let X' be a size-$(n-k)$ subset of X such that $\sigma[X']$ is suitable on $T[X]$. Then let γ be any order on Y: the concatenation $\sigma[X']\gamma$ is suitable both on $T_1[X' \cup Y]$ and $T_2[X' \cup Y]$,

so it is a valid solution for TTDE(T_1, T_2) of size $2n - k$, i.e., with k deletions.

$\Leftarrow$ Let X', Y' be subsets of X, Y, respectively, and σ' be an order on $X' \cup Y'$ such that σ' is suitable on both $T_1[X' \cup Y']$ and $T_2[X' \cup Y']$, and such that $|X' \cup Y'| \geq 2n - k > n$ (in particular, Y' contains at least one element denoted y, and $|X'| \geq n - k$). From T_2, it follows that σ' is the concatenation (in any order) of an order σ_x of X' suitable for $T[X']$ and an order σ_y of Y'. Assume first that σ_x appears before σ_y. Then consider each internal node r_i of the caterpillar T_1 with $i \leq n$ and a child c labelled with an element X'. Then this child must be ordered before all leaves below r_{i+1} since the corresponding subtree contains all leaves labelled with Y. Thus, the nodes in X' are ordered according to $\sigma[X']$, hence $\sigma_x = \sigma[X']$, and $T[X']$ is suitable with $\sigma[X']$. For the other case, where σ_y is ordered before σ_x, then for each r_i with a child in X', this child must be after the subtree with root r_{i+1} (containing Y), and the nodes in X' are ordered according to the reverse of $\sigma[X']$ (i.e., $\sigma_x = \overline{\sigma[X']}$). Thus, the reverse of $\sigma[X']$ is suitable for $T[X']$, and $\sigma[X']$ as well (this is obtained by reversing the permutation of all children of internal nodes of T). In both cases, X' is a solution for OTDE(T, σ) with $|X'| \geq n - k$. ◀

2.2 OTCM is NP-complete for trees with unbounded degree

▶ **Theorem 3.** *The OTCM problem is NP-complete for strict orders and therefore for weak orders.*

Proof. First note that OTCM is in NP, since, given an X-tree T with its leaves ordered according to an order σ' on X suitable on T, an order σ and a set L of leaves, the number of inversions between σ' and σ can be counted in $O(|L|^2)$. Regarding NP-hardness, we now give a reduction from Feedback Arc Set, which is NP-hard [13], to OTCM.

We consider an instance of the Feedback Arc Set problem, that is a directed graph $G = (V = \{v_1, \ldots, v_n\}, A)$ such that $|A| = m$ and an integer f.

We now define an instance of the OTCM problem, illustrated in Figure 3. The set X of leaf labels is $\{v_i^j \mid i \in [1..n], j \in [1..2m]\}$. We define the order $\sigma(G)$ in the following way. For each arc (v_i, v_j) of G, whose rank in the lexicographic order is k, we add to $\sigma(G)$ a k^{th} supplementary ordered sequence (which we will later call a "block" corresponding to this arc) $v_i^{2k-1} v_j^{2k-1} X_{i,j}^{2k-1} \overline{X}_{i,j}^{2k} v_i^{2k} v_j^{2k}$, where $X_{i,j}^{k'}$ is the ordered sequence of $v_{i'}^{k'}$ where i' ranges from 1 to n, excluding i and j, and $\overline{X}_{i,j}^{k'}$ is the reverse of $X_{i,j}^{k'}$ (i.e., the ordered sequence of $v_{i'}^{k'}$ where i' ranges from n down to 1, excluding i and j). The tree $T(G)$ is made of a root with n children v_1 to v_n, each v_i having $2m$ children, the leaves labeled by $v_i^{k'}$ for $k' \in [1..2m]$.

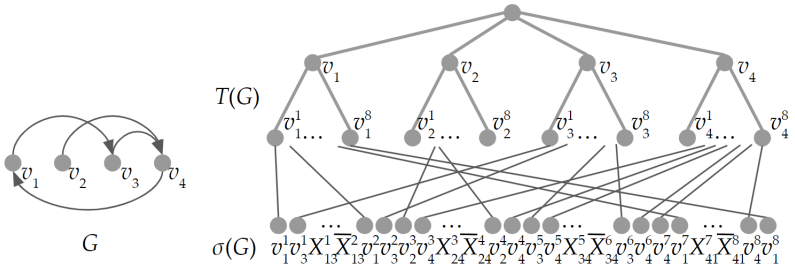


Figure 3 Illustration of the reduction of Feedback Arc Set to OTCM: a graph G with feedback arc set $S = \{(v_4, v_1)\}$ of size 1 and the corresponding tree $T(G)$ as well as the order $\sigma(G)$.

Given an ordering σ' suitable for T, and an inversion $(v_i^k, v_{i'}^{k'})$ forming an inversion between $\sigma(G)$ and σ', we say that this pair is *short-ranged* if $k = k'$, and *long-ranged*

otherwise. Furthermore, we say that σ' is *vertex-consistent* if, for every i and $k < k'$, we have $\sigma'(v_i^k) < \sigma(v_i^{k'})$. Finally, given σ', we write σ'' for the permutation of the $[1..n]$ corresponding to the children of the root.

We first claim that for any σ' suitable for T, there are at least $2\binom{n}{2}\binom{2m}{2}$ long-range inversions between σ' and $\sigma(G)$, and this bound is reached if σ' is vertex-consistent. Indeed, pick any pair $(v_i^k, v_{i'}^{k'})$ with $i \neq i'$ and $k \neq k'$. Then $v_i^k <_{\sigma(G)} v_{i'}^{k'}$ iff $k < k'$ (since they are in blocks k and k' of $\sigma(G)$), respectively, and $v_i^k <_{\sigma'} v_{i'}^{k'}$ iff $\sigma''(i) < \sigma''(i')$ (since they are in $L(T, v_i)$ and $L(T, v_{i'})$, respectively). Overall, among $4\binom{n}{2}\binom{2m}{2}$ such pairs of elements, there are $2\binom{n}{2}\binom{2m}{2}$ pairs creating an inversion (which is long-range by definition). For the case $i = i'$, note that pairs $(v_i^k, v_i^{k'})$ do not create any inversion iff σ' is vertex-consistent, which completes the proof of the claim.

Towards counting the number of short-ranged inversions, we say that an arc (v_i, v_j) of G is *satisfied* by σ'' if $\sigma''(i) < \sigma''(j)$. Let $i, j \in [1..n]$ and $k \in [1..m]$, and consider the two pairs (v_i^{2k-1}, v_j^{2k-1}) and (v_i^{2k}, v_j^{2k}). Then these two pairs are, by construction of T, in the same order in σ' (as defined by σ''). If the k^{th} arc of G is (v_i, v_j), then these two pairs are also in the same order in σ, i.e., together they account for either 0 or 2 (short-ranged) inversions. More precisely they yield 0 short-ranged inversions if (v_i, v_j) is satisfied by σ'', and 2 inversions otherwise. If the k^{th} arc of G is any other arc, then exactly one of $(v_i^{2k-1}, v_j^{2k-1}), (v_i^{2k}, v_j^{2k})$ forms a short-ranged inversion. Overall a pair $\{i, j\}$ such that one of $(v_i, v_j), (v_j, v_i)$ is a satisfied arc yields $m-1$ short-ranged inversions, a pair $\{i, j\}$ such that one of $(v_i, v_j), (v_j, v_i)$ is an unsatisfied arc yields $m+1$ short-range inversions, and any other pair $\{i, j\}$ with $i \neq j$ yields m short-ranged inversions. Overall, if there are f unsatisfied arcs, σ' yields $\binom{n}{2}m - m + 2f$ inversions.

We can now complete the proof with our main claim: G has a feedback arc set of size at most $f \Leftrightarrow$ the OTCM problem has a solution with at most $2\binom{n}{2}\binom{2m}{2} + \binom{n}{2}m - m + 2f$ inversions.

$\Rightarrow$: If G has a feedback arc set F of size f, as $G[A-F]$ is acyclic, we consider an order σ'' over n such that for all arcs (v_i, v_j) in $A - F$, $\sigma''(i) < \sigma''(j)$ (i.e., σ'' is the topological order of the vertices in $G[A-F]$). We now order the children v_i of the root of $T(G)$ according to this order σ'' and call σ' the induced order on the leaves of $T(G)$ (also sorting all leaves v_i^j below each v_i by increasing values of j). Note that σ' is vertex-consistent, and that an arc (v_i, v_j) is satisfied by σ'' iff $(v_i, v_j) \notin F$. Thus, σ' yields $2\binom{n}{2}\binom{2m}{2} + \binom{n}{2}m - m + 2f$ inversions.

$\Leftarrow$: Consider an order σ' suitable for T with at most $2\binom{n}{2}\binom{2m}{2} + \binom{n}{2}m - m + 2f$ inversions. Let σ'' be the corresponding order on the leaves of the root, and let F be the set of arcs unsatisfied by σ''. Since σ' has at least $2\binom{n}{2}\binom{2m}{2}$ long-range inversions, it has at most $\binom{n}{2}m - m + 2f$ short-range inversions, and $|F| \leq f$. Finally, since all arcs in $A - F$ are satisfied by σ'', $G[A-F]$ is acyclic and F is a feedback arc set. ◀

3 A polynomial-time algorithm for fixed-degree trees

We start by presenting a dynamic programming algorithm for fixed-degree trees, which is easy to implement and leads to an algorithm in $O(n^4)$ time for binary trees. The FPT algorithm presented in the next section has a better complexity but is more complex and reuses the dynamic programming machinery presented in this section, which explains why we start with this simpler algorithm.

▶ **Theorem 4.** *The OTDE problem can be solved in time $O(d!n^{d+2})$ for trees with fixed maximum degree d and for strict or weak orders.*

Proof. Given a vertex v of a rooted tree T, a (strict or weak) order $\sigma : L(T) \to [1..m]$ and two integers $l \leq r \in [1..m]$. We denote by $\mathcal{X}(v,l,r)$ a subset of $L(T,v)$ of maximum size such that $\sigma[\mathcal{X}(v,l,r)]$ is suitable with $T[\mathcal{X}(v,l,r)]$ and $\forall \ell \in \mathcal{X}(v,l,r), \sigma(\ell) \in [l,r]$. Note that $\mathcal{X}(v,l,r)$ also depends on T and σ but we simplify the notation by not mentioning them as they can clearly be identified from the context.

Denoting by $c_1, \ldots, c_k$ the children of v in T, we claim that the following formula allows to recursively compute $\mathcal{X}(v,l,r)$ in polynomial time:

- $|\mathcal{X}(v,l,r)| = \max_{\substack{\text{permutation } \pi \text{ of } [1..k] \\ x_1 = l \leq x_2 \leq \ldots \leq x_k \leq x_{k+1} = r}} \sum_{i=1}^{k} \left|\mathcal{X}(c_{\pi(i)}, x_i, x_{i+1})\right|$ if v is an internal node of T;
- for any leaf ℓ of T, $|\mathcal{X}(\ell,l,r)| = 1$ if $\sigma(\ell) \in [l,r]$, 0 otherwise.

Correctness: We prove by induction on the size of $L(v)$ that $\mathcal{X}(v,l,r)$ is indeed a subset of $L(T,v)$ of maximum size such that $\sigma[\mathcal{X}(v,l,r)]$ is suitable with $T[\mathcal{X}(v,l,r)]$ and $\forall \ell \in \mathcal{X}(v,l,r), \sigma(\ell) \in [l,r]$.

This is obvious for any leaf, so let us consider a vertex v of T with a set $\{c_1, \ldots c_k\}$ of children. Suppose for contradiction that there exists a set of integers $l \leq r$ and a subset X' of $L(v)$ of size strictly greater than $\mathcal{X}(v,l,r)$ such that $\sigma[X']$ is suitable with $T[X']$ and $\forall \ell \in X', \sigma(\ell) \in [l,r]$. We then denote by $X'_1, \ldots$ and X'_k the sets of leaves $L(c_1) \cap X', \ldots$ and $L(c_k) \cap X'$, respectively. Without loss of generality we consider that the children c_i of v are labeled such that $\max_{\ell \in X'_i}\{\sigma(\ell)\} \leq \min_{\ell \in X'_{i+1}}\{\sigma(\ell)\}$. For all $i \in [2..k]$, we define $m_i = \min_{\ell \in X'_i}\{\sigma(\ell)\}$, $m_1 = l$ and $m_{k+1} = r$. Using the induction hypothesis we know that for each $i \in [1..k]$, $|X'_i| \leq \left|\mathcal{X}(v, \min_{\ell \in X'_i}\{\sigma(\ell)\}, \max_{\ell \in X'_i}\{\sigma(\ell)\})\right|$, so $|X'_i| \leq |\mathcal{X}(v, m_i, m_{i+1})|$ because $\left[\min_{\ell \in X'_i}\{\sigma(\ell)\}, \max_{\ell \in X'_i}\{\sigma(\ell)\}\right] \subseteq [m_i, m_{i+1}]$. Therefore, $|X'| = \sum_{i=1}^{k} |X'_i| \leq \sum_{i=1}^{k} |\mathcal{X}(v, m_i, m_{i+1})|$ so by definition of $\sigma[\mathcal{X}(v,l,r)]$, $|X'| \leq \sigma[\mathcal{X}(v,l,r)]$: contradiction!

We therefore obtain a correct solution of $OTDE(T,\sigma)$ by computing $\mathcal{X}(\text{root}(T), 0, m)$.

Running-time: For each v, we compute the table of the $O(n^2)$ values of $\mathcal{X}(v,l,r)$ for all intervals $[l,r]$. Each of these values can be computed by generating the $k!$ permutations of children of v to consider any possible order among the children and splitting the interval $[l,r]$ into any possible configurations of d consecutive intervals with integer bounds partitioning $[l,r]$, which can be done in time $O(n^{d-1})$. So the computation of each $\mathcal{X}(v,l,r)$ is done in time $O(d!n^{d-1})$, therefore the total computation of all $\mathcal{X}(v,l,r)$ is done in time $O(n \times n^2 \times d!n^{d-1})$, that is in $O(d!n^{d+2})$. ◀

4 An FPT algorithm for the *deletion-degree* parameter for OTDE

We recall that with a reduction of OTDE to 3-Hitting Set [10], using the best algorithm known so far to solve this problem[4], we can obtain an algorithm to solve OTDE $O^*(2.08^k)$ [23], where k is the number of leaves to delete and the O^* notation ignores the polynomial factor. In this section we obtain an FPT algorithm in time $O(n^4 d \partial 2^{\partial})$, where d is the maximum degree of the tree and ∂ is the deletion-degree of the solution.

▶ **Theorem 5.** *The OTDE problem parameterized by the deletion-degree ∂ of the solution is FPT and can be solved in time $O(n^4 d \partial 2^{\partial})$ for strict or weak orders.*

We adapt the dynamic programming algorithm from Theorem 4, using a vertex cover subroutine to have a good estimation of the permutation of the children of each node.

[4] http://fpt.wikidot.com/fpt-races

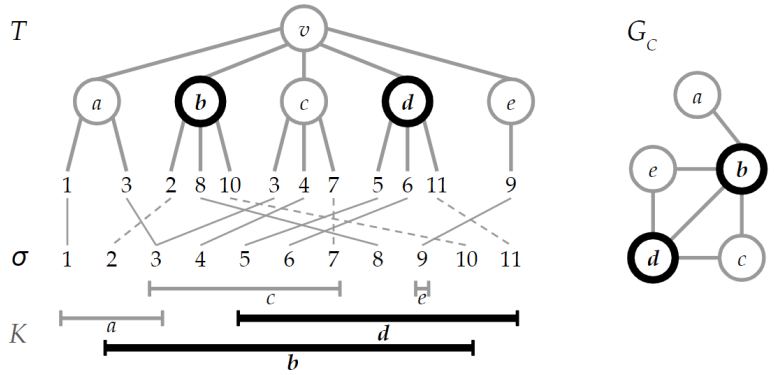


Figure 4 An instance (T, σ) of OTDE (top-left), with a vertex v having children set $C_v = \{a, b, c, d, e\}$. The conflict graph of C_v (right) has a size-2 vertex cover $K = \{b, d\}$. Based on the span of each vertex (bottom-right), the dynamic programming algorithm tests permutations of C_v such that (a, c, e) appear in this order, interleaved in any possible way with b and d. In particular, the final solution corresponds to the permutation $(a\ c\ d\ b\ e)$ of C_v. Note that since σ may be a weak order (two leaves are labelled 3 in the example), the conflict graph does not correspond exactly to the intersection graph of the span intervals, e.g. vertices a and c are *not* in conflict, even though their spans overlap.

We first introduce some definitions (see Figure 4 for a illustration of these definitions and the algorithm in general). Given any vertex v of T, let C_v be the (independent) set of children of v, and let G_v be the *conflict graph* with vertex set C_v and with one edge per conflict. Let K be a vertex cover of G_v. Then the vertices of $C_v \setminus K$ have a *canonical order* $(w_1, \ldots, w_{k'})$, with $k' = |C_v \setminus K|$ and $w_i \preceq_\sigma w_j$ for all $i \leq j$ (ties may happen when two children contain a single leaf each which are equal, such ties are broken arbitrarily). We say that $P \subseteq C_v$ is a *prefix of* C_v *wrt.* K if $P \setminus K$ is a prefix of this order (i.e., for some $i \leq k'$, $P \setminus K = \{w_1, \ldots, w_i\}$). In other words, ignoring all subtrees below vertices of K, all leaves below vertices of a prefix P are necessarily ordered before leaves below vertices outside of P.

▶ **Lemma 6.** *If X' is a solution of OTDE with deletion-degree ∂, then for any vertex v of T, the conflict graph G_v admits a vertex cover of size at most ∂.*

Proof. Given a subset X' of X, we say that a node v of T *has a deletion* if some $L(v) \not\subseteq X'$, i.e., if v has a leaf in $X \setminus X'$. Let $\{u, v\}$ be any conflict (edge) of the conflict graph G_v, then at least one of u, v has a deletion for X' (indeed, the conflict involves three leaves a, b, c, of which at least one must be deleted). Thus, the vertices with a deletion in G_v form a vertex cover of this graph. The lemma follows from the fact that at most ∂ vertices have a deletion in each conflict graph. ◀

The first step of our algorithm consists in computing, for each node v of the graph, the set C of children of v, its conflict graph G_v, and a minimum vertex cover K_v of G_C. Since each K_v has size at most ∂ (by Lemma 6), K_v can be computed in time $O(1.3^\partial + \partial n)$ [5], and overall this first step takes $O(1.3^\partial n + \partial n^2)$.

We proceed with the dynamic programming part of our algorithm. To this end, we generalize the table $\mathcal{X}$ to sets of nodes (instead of only v) as follows: $\mathcal{X}(P, l, r)$ corresponds to the largest set X of leaves in $\bigcup_{u \in P} L(u)$ such that σ_X is suitable for $T[X]$. Note that for a node v with children set C, $\mathcal{X}(v, l, r) = \mathcal{X}(\{v\}, l, r) = \mathcal{X}(C, l, r)$.

We first compute $\mathcal{X}(\{v\}, l, r)$ for each leaf v: clearly $\mathcal{X}(\{v\}, l, r) = \{u\}$ if $l \leq \sigma(v) \leq r$, and $\mathcal{X}(\{v\}, l, r) = \emptyset$ otherwise. For each internal vertex v (visiting the tree bottom-up), we

obtain $\mathcal{X}(\{v\}, l, r)$ by first computing $\mathcal{X}(P, l, r)$ for each prefix P of C_v by increasing order of size, using the following formulas:

$$|\mathcal{X}(P, l, r)| = \emptyset \text{ if } P = \emptyset$$
$$= \max_{\substack{x \in [l..r],\ u \in P \\ P \setminus \{u\} \text{ prefix of } C_v}} |\mathcal{X}(P \setminus \{u\}, l, x)| + |\mathcal{X}(\{u\}, x, r)|$$
$$|\mathcal{X}(\{v\}, l, r)| = |\mathcal{X}(C_v, l, r)|$$

Each vertex v has at most $d2^\partial$ prefixes, so the dynamic programming table $\mathcal{X}$ has at most $n^3 d 2^\partial$ cells to fill. For each prefix P, there exist at most $\partial + 1$ vertices $u \in P$ such that $P \setminus \{u\}$ is a prefix (u can be any vertex in $P \cap K_v$, or the maximum vertex for $\preceq_\sigma$ in $P \setminus K_v$). Overall, the *max* is taken over $O(n\partial)$ elements, and $\mathcal{X}$ can be filled in time $O(n^4 d \partial 2^\partial)$.

Before proving the correctness of the above formula, we need a final definition: given a set of leaves $X' \subseteq X$ and a vertex v of T, we write $\text{span}_{X'}(v)$ for the smallest interval containing $\sigma(u)$ for each leaf $u \in L(u) \cap X'$ (note that $\text{span}_{X'}(v)$ may be empty, if all its leaves are deleted in X').

▶ **Lemma 7.** *Let X' be a solution of OTDE(T, σ), $v \in T$ and $1 \leq l \leq r \leq m$ such that $\text{span}_{X'}(v) \subseteq [l, r]$. Then there exists a permutation $(c_1 \ \ldots \ c_k)$ of the children of v and integers $x_0 = l \leq x_1 \leq \ldots \leq x_k = r$ such that, for each $i \leq k$,*

(a) $\text{span}_{X'}(c_i) \subseteq [x_{i-1}, x_i]$, and

(b) $P_i = \{c_1, \ldots, c_i\}$ is a prefix of the children of v wrt. σ.

Proof. Recall that we write C_v and K_v, respectively, for the set of chidren of v and the vertex cover in the conflict graph induced by these children. For each element c of C_v with a non-empty span, let $x(c) = \max(\text{span}(c))$. For each element w_i of $C_v \setminus K_v$ with an empty span (taking i for the rank according to the canonical order), let $x(w_i) = x(w_{i-1})$ (and $x(w_1) = l$ for $i = 1$). For the remaining vertices (in K_v with an empty span), set $x(c) = l$. Finally, order vertices $c_1, \ldots, c_k$ by increasing values of $x(c_i)$ (breaking ties according to the canonical order when applicable, or arbitrarily otherwise), and set $x_i = x(c_i)$.

Condition (a) follows from the fact that X' is a solution for OTDE(T, σ), so that the span covered by the leaves under siblings do not overlap. For condition (b) we refer to the definition of prefix: each $P_i \setminus K_c$ is indeed a prefix in the canonical ordering of $C_v \setminus K_v$. ◀

The dynamic programming formula follows from the above remark: one can build the solution by incrementing prefixes one vertex at a time (rather than trying all possible permutations of children, as in Theorem 4).

5 Optimizing OTCM and OTDE are two different things

In order to ensure that finding the smallest k such that OTCM or OTDE outputs a positive answer actually consists in optimizing different criteria, we provide in Figure 5 an example of X-tree and an order of its leaves where the order reaching the best k for a positive answer of the OTCM problem does not provide the optimal value for the number of leaves to delete in a positive answer of OTDE and where the best k for a positive answer of the OTDE problem does not provide an optimal value for the number of inversions for a positive answer of the OTCM problem.

We checked the optimality for both criteria by implementing the "naive" dynamic programming $O(n^2)$ algorithm described in Section 2.1 of [10] to solve the OTCM problem and the $O(n^4)$ algorithm described in Section 3 to solve the OTDE problem on binary trees.

Both implementations are available in Python, under the GPLv3 licence, at `https://github.com/oseminck/tree_order_evaluation`, as well as the file `inputCounterExample1b.txt` containing the Newick encoding for the tree of Figure 5.

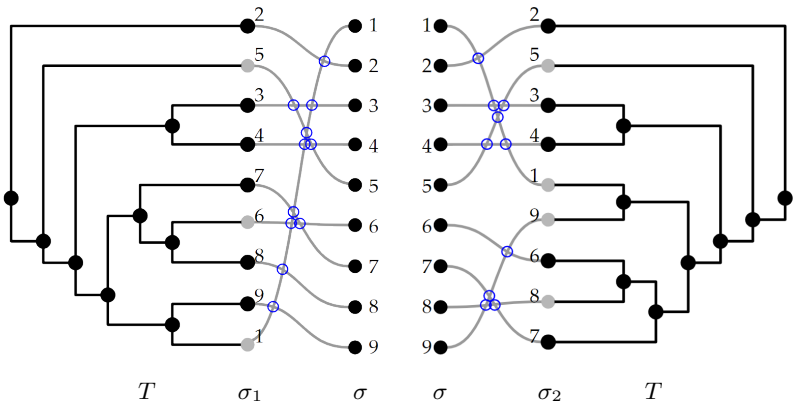


Figure 5 Two planar embeddings of a rooted tree T: the one on the left is optimal for the OTDE problem (deleting the 3 gray leaves makes the order σ suitable on T restricted to the remaining leaves, but the order σ_1 suitable on T has 11 inversions, shown with empty circles, with σ); the other one is optimal for the OTCM problem with the order σ_2 suitable on T having 10 inversions with σ but not for the OTDE problem (4 leaves, for example the 4 gray ones, need to be deleted to make the order σ suitable on T restricted to the remaining leaves).

6 Experiments and discussion

In this section, we investigate the potential for use of OTCM and OTDE in applications where the tree of elements is obtained from a clustering algorithm taking as input distances between those elements, and where we want to test whether this clustering reflects some intrinsic order on the elements, for example the chronological order. We both test the running time of OTCM and OTDE on real data, and the performance of OTDE on simulated data to detect possibly misplaced leaves in the order.

The first experiment deals with text data: the CIDRE corpus [20] that contains the works of 11 French 19$^{\text{th}}$ century fiction writers dated by year (every file contains a book that is annotated with its year of writing). We apply apply hierarchical clustering on the different corpora using the `AgglomerativeClustering` class from the package `sklearn` [18]. Distance matrices on which the clustering is based are obtained by using the relative frequencies of the 500 most frequent tokens[5] in each corpus. Distance matrices were generated using the R package stylo [8], with the *canberra* distance metric. We obtain the results given in Table 1, which provides the running time in milliseconds of the algorithms we implemented to solve OTCM and OTDE. They show that both algorithms on binary trees are quick enough to handle typical instances of the OTCM and the OTDE problems relevant for digital humanities, a few milliseconds for the first one and a few seconds for the second one, for

[5] A token is (a part of) a word form or a punctuation marker. The last sentence would yield the following tokens: [“A”, “token”, “is”, “(”, “a”, “part”, “of”, “)”, “a”, “word”, “form”, “or”, “a”, “punctuation”, “marker”, “.”] Deliberately, we do not use the term “word”, because the word can be seen as a linguistic unit of form and meaning, and henceforward “punctuation marker” would be one word and the period in the end of the sentence would not be one.

tree	# leaves	OTCM time	# inversions	p_{OTCM}	OTDE time	# deleted leaves	p_{OTDE}
Ségur	22	1	40	0.24	200	9	1
Féval	23	2	47	0.38	268	8	0
Aimard	24	1	35	0	401	8	0
Lesueur	31	1	48	0	676	13	0
Zévaco	29	1	42	0	727	11	0
Zola	35	2	60	0	1203	9	0
Gréville	36	2	105	0	2211	18	1
Ponson	42	3	167	2.23	3447	18	0
Balzac	59	4	248	0	8292	34	0
Verne	58	3	183	0	13446	27	0
Sand	62	4	283	0	17557	39	1

Table 1 Results of our implementations for problems OTCM and OTDE on binary trees generated from corpora of French novels of the 19$^{\text{th}}$ century. Time durations are given in milliseconds.

instances of about 50 elements in the tree and in the order.

Investigating precisely whether the numbers of inversions or deleted leaves shown in Table 1 are sufficiently small to reflect consistency with a chronological signal is beyond the scope of this paper. However, we also provide p_{OTCM} and p_{OTDE}, the percentage of cases when the best order on the leaves of the tree has the same number of inversions, or less than the chronological order, among 10000 randomly generated orders for OTCM and 100 randomly generated orders for OTDE, respectively[6]. These numbers illustrate that in all cases, it is unlikely that the observed optimal numbers of inversions or deleted leaves are due to chance, as we get equal or smaller values of inversions or deleted leaves on less than 3% of random orders (for Ponson du Terrail the number of inversions is 167 or less for 2.23% of random orders; for one of the 10 000 simulated random orders, it reached as little as 124 inversions). These preliminary results obtained thanks to reasonably small running times open new perspectives in investigating further the practical use of these algorithms, and comparing their results with other methods to search for signals of chronological evolution in textual data [21].

Our second experiment involves simulated data, to check whether, in the case the tree is built to be consistent with the input order, our algorithm finding the minimum of leaves in the tree to remove inconsistencies with the order is able to detect errors that we intentionally add to the order. We produced 100 instances of the OTDE problem, for each chosen value of n, the number of leaves, and $e < n$, the number of errors, in the following manner:

1. we randomly pick n distinct integers from the interval $[0, 999]$, which will be our set X of leaves;
2. we build a distance matrix in which the distance between two elements from X is simply the absolute difference between both; we add some noise to this matrix by adding or subtracting in each cell a random quantity equal to at most 10% of the cell value, obtaining a noisy matrix, from which we build an X-tree T using the `AgglomerativeClustering` class from the package `sklearn`;
3. we randomly pick a set L_e of e leaves in X and replace their value by another integer,

[6] We chose to generate less random orders for OTDE in our simulations, as our algorithm is slower to solve this problem than OTCM.

n = # leaves	e = # errors	proportion of cases when $L = L_e$	when $\lvert L - L_e \rvert = 1$
20	1	0.79	1
20	2	0.62	0.96
20	3	0.39	0.88
20	4	0.33	0.77
20	5	0.27	0.67
50	1	0.93	1
50	2	0.83	0.99
50	3	0.70	0.98
50	4	0.59	0.91
50	5	0.56	0.90

Table 2 Results of the attempts to perfectly detect the set L_e of randomly relabeled leaves in simulated trees (when $L = L_e$); the situation when $|L - L_e| = 1$ corresponds to finding only $e - 1$ leaves among the e randomly relabeled leaves).

randomly chosen from the interval $[0, 999]$, distinct from other leaf labels; σ is the set of leaves ordered by increasing value taking into account these new values;

4. by solving the OTDE problem on T and σ, we compute the minimum set L of leaves to remove to make $\sigma[X - L]$ suitable on $T[X - L]$, and check whether $L = L_e$.

This experiment simulates the situation where we would have dating errors on the elements we clustered in a tree. Note that like in the case of dating errors, the error in our simulation may not change the overall order on the leaves. Table 2 provides, for each chosen values of n and e, the proportion of simulated instances of $OTDE$ where $L = L_e$, that is when our algorithm removed exactly the e leaves whose label had been randomly modified. We can observe that this happens in a majority of cases only when the number of modified leaves is small compared with the total number of leaves (up to 2 for 20 leaves, up to 4 for 50 leaves). Solving OTDE still allows to identify $e - 1$ among the e modified leaves in a majority of cases in all our experiments.

7 Conclusion and perspectives

In this article, we addressed two problems initially introduced with motivations from bioinformatics, OTCM and OTDE. We stated them in a more simple framework with a tree and an order as input, instead of two trees as was the case when they were introduced, opening perspectives for new practical uses in digital humanities and proving that they are not equivalent. We proved that both problems, as well as a problem on two trees, TTDE, are NP-complete in the general case. We gave a polynomial-time algorithm for OTDE on trees with fixed maximum degree and an FPT algorithm in a parameter possibly smaller than the size of the solution for arbitrary trees.

We also investigated their potential for practical use, checking that the algorithms we implemented with open source code in Python to solve them are well suited for applications in digital humanities in terms of running time. We also observed on simulated data that it is possible to identify a small number of leaves for which there would be an ordering error if the tree is built from distance data derived from an order on its leaves. Future research includes the search for FPT algorithms, with relevant parameters, for OTCM and TTDE.

References

1. Mukul S Bansal, Wen-Chieh Chang, Oliver Eulenstein, and David Fernández-Baca. Generalized binary tanglegrams: Algorithms and applications. In *International Conference on Bioinformatics and Computational Biology*, pages 114–125. Springer, 2009. doi:10.1007/978-3-642-00727-9_13.
2. Ziv Bar-Joseph, Erik D. Demaine, David K. Gifford, Nathan Srebro, Angèle M. Hamel, and Tommi S. Jaakkola. K-ary clustering with optimal leaf ordering for gene expression data. *Bioinformatics*, 19(9):1070–1078, 2003. doi:10.1093/bioinformatics/btg030.
3. Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl 1):S22–S29, 2001. doi:10.1093/bioinformatics/17.suppl_1.S22.
4. Ulrik Brandes. Optimal leaf ordering of complete binary trees. *Journal of Discrete Algorithms*, 5(3):546–552, 2007. doi:10.1016/j.jda.2006.09.003.
5. Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.
6. Tim Dwyer and Falk Schreiber. Optimal leaf ordering for two and a half dimensional phylogenetic tree visualisation. In *APVis '04: Proceedings of the 2004 Australasian symposium on Information Visualisation*, volume 35, pages 109–115, 2004. doi:10.5555/1082101.1082114.
7. Denise Earle and Catherine B. Hurley. Advances in dendrogram seriation for application to visualization. *Journal of Computational and Graphical Statistics*, 24(1):1–25, 2015. doi:10.1080/10618600.2013.874295.
8. Maciej Eder, Jan Rybicki, and Mike Kestemont. Stylometry with R: a package for computational text analysis. *R Journal*, 8(1):107–121, 2016. URL: https://journal.r-project.org/archive/2016/RJ-2016-007/index.html.
9. Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing minimization. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 457–469. Springer, 2005. doi:10.1007/11590156_37.
10. Henning Fernau, Michael Kaufmann, and Mathias Poths. Comparing trees via crossing minimization. *Journal of Computer and System Sciences*, 76(7):593–608, 2010. doi:10.1016/j.jcss.2009.10.014.
11. Philippe Gambette, Olga Seminck, Dominique Legallois, and Thierry Poibeau. Evaluating hierarchical clustering methods for corpora with chronological order. In *EADH2021: Interdisciplinary Perspectives on Data. Second International Conference of the European Association for Digital Humanities*, Krasnoyarsk, Russia, September 2021. EADH. URL: https://hal.archives-ouvertes.fr/hal-03341803.
12. Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: an introduction to the R package seriation. *Journal of Statistical Software*, 25(3):1–34, 2008. doi:10.18637/jss.v025.i03.
13. Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.
14. Cyril Labbé and Dominique Labbé. Existe-t-il un genre épistolaire? Hugo, Flaubert et Maupassant. In *Nouvelles Journées de l'ERLA*, pages 53–85. L'Harmattan, 2013.
15. Jean-Marc Leblanc. *Analyses lexicométriques des vœux présidentiels*. ISTE Group, 2016.
16. Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001. doi:10.1006/jctb.2000.2026.
17. Hermann Moisl. How to visualize high-dimensional data: a roadmap. *Journal of Data Mining & Digital Humanities*, 2020. doi:10.46298/jdmdh.5594.
18. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

527 **19** Ryo Sakai, Raf Winand, Toni Verbeiren, Andrew Vande Moere, and Jan Aerts. dendsort:
528 modular leaf ordering methods for dendrogram representations in R. *F1000Research*, 3, 2014.
529 doi:10.12688/f1000research.4784.1.
530 **20** Olga Seminck, Philippe Gambette, Dominique Legallois, and Thierry Poibeau. The corpus for
531 idiolectal research (CIDRE). *Journal of Open Humanities Data*, 7:15, 2021. doi:10.5334/
532 johd.42.
533 **21** Olga Seminck, Philippe Gambette, Dominique Legallois, and Thierry Poibeau. The evolution
534 of the idiolect over the lifetime: A quantitative and qualitative study on French 19^{th} century
535 literature, 2022. Under review.
536 **22** Balaji Venkatachalam, Jim Apple, Katherine St John, and Dan Gusfield. Untangling tan-
537 glegrams: comparing trees by their drawings. *IEEE/ACM Transactions on Computational*
538 *Biology and Bioinformatics*, 7(4):588–597, 2010. doi:10.1109/TCBB.2010.57.
539 **23** Magnus Wahlström. *Algorithms, measures and upper bounds for satisfiability and related prob-*
540 *lems*. PhD thesis, Department of Computer and Information Science, Linköpings universitet,
541 2007. URL: http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Aliu%3Adiva-8714.

Automatic Normalisation of Early Modern French

Rachel Bawden, Jonathan Poinhos, Eleni Kogkitsidou, Philippe Gambette, Benoît Sagot, Simon Gabay

Automatic Normalisation of Early Modern French

Rachel Bawden[1] **Jonathan Poinhos**[2] **Eleni Kogkitsidou**[2]
Philippe Gambette[2] **Benoît Sagot**[1] **Simon Gabay**[3]
[1]Inria, Paris, France
[2]LIGM (UMR 8049), Université Gustave Eiffel, CNRS, 77454 Marne-la-Vallée, France
[3]Université de Genève, Switzerland
firstname.lastname@{inria.fr,univ-eiffel.fr,unige.ch},

Abstract

Spelling normalisation is a useful step in the study and analysis of historical language texts, whether it is manual analysis by experts or automatic analysis using downstream natural language processing (NLP) tools. Not only does it help to homogenise the variable spelling that often exists in historical texts, but it also facilitates the use of off-the-shelf contemporary NLP tools, if contemporary spelling conventions are used for normalisation. We present FREEM$_{norm}$, a new benchmark for the normalisation of Early Modern French (from the 17th century) into contemporary French and provide a thorough comparison of three different normalisation methods: ABA, an alignment-based approach and MT-approaches, (both statistical and neural), including extensive parameter searching, which is often missing in the normalisation literature.

Keywords: Digital Humanities, Normalisation, Spelling, Modern French, Machine Translation, Historical

1. Introduction

Computational approaches have recently been playing an increasing role in the humanities (Gabay, 2021), especially concerning the study of textual documents. Historical documents are particularly interesting, as they are an invaluable source of historical information and are crucial witnesses of language evolution.

Whether documents are to be studied manually by philologists and literary experts or analysed automatically using downstream natural language processing (NLP) tasks such as part-of-speech (PoS) tagging and parsing, a useful preliminary step is *normalisation*, which consists in modernising the spelling of the documents to conform to contemporary spelling conventions. Normalisation has the effect of (i) reducing spelling variation present in historical documents, often written at a time spelling was not standardised, and (ii) reducing the gap between the historical state of the language and the contemporary state. Importantly, this allows us to apply off-the-shelf NLP tools to old texts and limit the performance drop that can usually be expected, for example for tagging and parsing (Pettersson et al., 2013b) or geographical named entity recognition (Kogkitsidou and Gambette, 2020).

There has been a considerable amount of previous research in historical spelling normalisation, with a range of methods being developed, including manually developed rules (Porta et al., 2013; Baron and Rayson, 2009; Riguet, 2019), those exploiting edit distances and other external resources such as lexicons (Mitankin et al., 2014) and machine translation (MT) approaches, both statistical (Scherrer and Erjavec, 2013; Domingo and Casacuberta, 2018a) and neural (Bollmann and Søgaard, 2016; Hämäläinen et al., 2018). Despite this, questions still remain regarding which method is the most effective, particularly between statistical MT (SMT) and neural MT (NMT) approaches. There has for example been little research in optimising these models for the particular task, which could lead to false conclusions being drawn about which model is best; as has been previously shown for low-resource tasks, neural models in particular are sensitive to model size, training parameters and the degree of subword segmentation applied to texts (Sennrich and Zhang, 2019; Fourrier et al., 2021).

Our focus in this paper is on the normalisation into contemporary French of Early Modern French (also known as Modern French or Classical French), which is French from the 17th century. Despite several recent efforts (Gabay and Barrault, 2020; Gabay et al., 2019; Riguet, 2019), there has so far been very little research carried out on spelling normalisation for historical French, and so we aim to fill this gap. Figure 1 illustrates a few of the normalisation types observed, from simple typographic changes (e.g. *ſ* → *s*), changes to segmentation (*long temps* 'a long time' → *longtemps*), changes reflecting language change (*eſtoit* '(s/he) was' → *était*) and the use of classical false etymological spellings (e.g. *ç* being used in Modern French *ſçavoir* 'to know' as a link to Latin *scire*, from which it does not originate).

In this paper, we present the parallel normalisation corpus, FREEM$_{norm}$ (for Early Modern French), on which we train and evaluate, and, in addition to baseline models, we compare three methods: (i) an alignment-based approach, called ABA, using automatically learned word correspondences from a parallel corpus, (ii) phrase-based SMT, and (iii) NMT, comparing an LSTM model (Bahdanau et al., 2015) and a Transformer (Vaswani et al., 2017). We find that despite extensive parameter optimisation for NMT models, SMT produces the best results overall, with all methods largely exceeding the baselines. Our comparison shows that the methods exhibit quite different be-

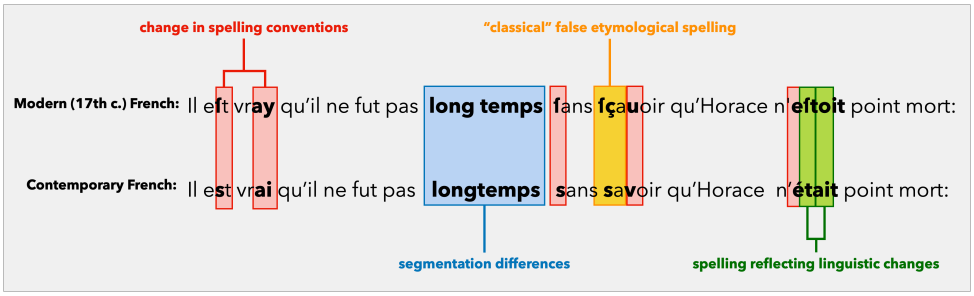


Figure 1: A Modern French sentence and its contemporary French normalisation.

haviour in terms of how conservative or inventive they are, which could be useful information depending on the downstream task (e.g. as a pre-annotation tool for manual annotation or a downstream NLP application). Our main contributions can be summarised as follows:

- Introduction of a new benchmark for the normalisation of Modern French, which can be used in further research.
- Extensive experiments comparing an alignment-based approach (ABA) with three MT approaches (SMT, LSTM and Transformer), with best results achieved by SMT. We also show that a lexicon-based post-processing step can systematically improve over all other methods tested.
- We freely distribute the data,[1] scripts and state-of-the-art normalisation models.[2,3]

2. Related Work

A considerable amount of work has been carried out in historical spelling normalisation, across various languages, with research dating back to the 1980s (Fix, 1980). A range of different approaches have been developed, including rule-based (Porta et al., 2013; Riguet, 2019), the use of various types of edit-distance (Hauser and Schulz, 2007; Bollmann, 2012; Pettersson et al., 2013a) and MT-style approaches, both statistical (Vilar et al., 2007; Scherrer and Erjavec, 2013; Ljubesic et al., 2016; Domingo et al., 2017) and neural (Korchagina, 2017; Domingo and Casacuberta, 2018b; Tang et al., 2018). Interestingly, all of these approaches remain useful today, thanks to their different strengths, depending on the type of normalisation and the amount of data available (Bollmann, 2019).

2.1. Word Lists, Rules and Edit-based Methods

Approaches relying on word lists, consisting in simply replacing historical variants by their normalised equivalent have been developed in several languages: English (Reynaert et al., 2012), German, Portuguese (Piotrowski, 2012) and Slovene (Erjavec et al., 2011).

Many rule-based and edit-distance-based approaches are unsupervised (i.e. they do not require parallel data), which is a considerable advantage, especially for historical varieties for which annotated data is not readily available. Rules can be developed manually by experts (Porta et al., 2013; Baron and Rayson, 2009; Riguet, 2019) or be extracted from a comparison of historical and modern word lists or parallel data if this is available (Bollmann et al., 2011).

The use of edit distance, using for example Levenshtein distance is often a strong baseline (Pettersson et al., 2013a), due to the fact that the surface forms of historical and contemporary spellings are often very similar and the alignment between both words and characters in the two varieties is almost perfectly monotonic. Basic edit-distance can be enhanced with specific weights for different edits (Bollmann et al., 2011) or based on characters or character groups (Hauser and Schulz, 2007; Bollmann, 2012), given the observation that certain errors are more serious than others.

2.2. Normalisation as MT

MT approaches to the problem have been popular, with the historical and modern states of the language being treated as the source and target languages respectively.

Characters, Subword or Words? Most previous research has focused on character-based MT, which models transformations at the level of individual characters (Vilar et al., 2007; Scherrer and Erjavec, 2013; Pettersson et al., 2013b; Domingo and Casacuberta, 2021), which makes sense for the task of spelling normalisation, as it often involves local transformations and largely monotonic alignments between source and target sentences. However, there has since been work exploring word translation, subword translation (Tang et al., 2018) or a mixture of these (Vilar et al., 2007; Domingo and Casacuberta, 2021). It is rare however for works in historical spelling normalisation to explore the optimal degree of segmentation, although Tang et al. (2018) do find subwords to be more effective than character-based: character-based segmentation offers a greater possibility for generalisation with the caveat that it requires the model to learn to translate longer sequences and learn patterns better, whereas word or subword segmentation can exploit models' ability to

[1] https://doi.org/10.5281/zenodo.5865428
[2] https://github.com/rbawden/ModFr-Norm
[3] See https://freem-corpora.github.io for the project page.

memorise, but may run the risk of limited generalisation, especially to unseen or less frequent words.

SMT or NMT? The first approaches were with SMT (Koehn et al., 2007), which proved more effective than rule-based and edit-distance based approaches (Pettersson et al., 2014; Hämäläinen et al., 2018; Bollmann, 2019), when there is parallel data available, and even when this data is produced synthetically (Scherrer and Erjavec, 2013; Domingo and Casacuberta, 2018a). NMT approaches to historical spelling normalisation were developed as it took off in the domain of general MT (Bollmann and Søgaard, 2016; Hämäläinen et al., 2018). Comparisons between SMT and NMT show different results, with SMT being superior in some cases (Domingo and Casacuberta, 2018a), and NMT in others (Bollmann, 2019), provided enough parallel data is available (Bollmann, 2019). Importantly, the methods appear to have different behaviours and therefore their own strengths and weaknesses, meaning that a single method (including rule-based approaches) is not necessarily a systematically better choice (Hämäläinen et al., 2018; Robertson and Goldwater, 2018).

Word Translation vs. Sentence Translation A considerable portion of the research in historical normalisation is based on the normalisation of word lists, so of words in isolation. However, as discussed in (Ljubesic et al., 2016), it can be beneficial in some contexts to normalise whole sentences (where there is ambiguity in the normalised form that should be chosen). This has the disadvantage of creating longer sequences to process, but is necessary in order to hope to handle all phenomena. The development of parallel corpora rather than word lists has encouraged research in this direction (Tjong Kim Sang et al., 2017; Gabay and Barrault, 2020; Ortiz Suarez et al., 2022).

2.3. Normalisation for Historical French

Despite there being a plethora of research on historical spelling normalisation, little research has been done so far on historical French, with most work focusing on Dutch, German, Hungarian, Slovene, and Swedish, helped by the existence of benchmark data (Dipper and Schultz-Balluff, 2013) and shared tasks (Ljubesic et al., 2016; Tjong Kim Sang et al., 2017).

A collaborative word list associating normalised versions of historical words in French was started in 2009 on the Wikisource digital library,[4] which is available for automatic normalisation through word substitution (The French Wikisource Community, 2022). Recently, there has been some preliminary research, with the development of a parallel corpus for the normalisation of Modern French (from the 17th c.) (Ortiz Suarez et al., 2022) and first baselines, including rule-based (Riguet, 2019) and NMT-style approaches (Gabay et al., 2019; Gabay and Barrault, 2020). Gabay and Barrault (2020) compare character-based SMT and NMT at different granularities (words, subwords and characters): NMT outperformed SMT, and for NMT, the best input representations were found to be words, then characters, then subwords. However, they do not seem to perform a comparison of different levels of subword segmentation or of different sizes of architecture, which has been shown to be important when drawing conclusions about the usability of NMT in low-resource settings (Sennrich and Zhang, 2019).

[4]On 17th January 2022, the whole list contains 15,470 words and expressions and their normalised equivalents.

3. Approaches Compared

We present and compare several approaches, representing a wide range of techniques: (i) an alignment-based method using a parallel corpus (Section 3.1), (ii) statistical MT (Section 3.2.1), (iii) neural MT, testing both LSTM and Transformer models (Section 3.2.2). In addition to comparing these approaches to two baselines described in Section 6.1, we also assess the impact of a lexicon-based post-processing described in Section 3.3.

3.1. ABA: Alignment-based

The ABA method (short for alignment-based method), is a hybrid approach consisting of (i) word-level transformation rules that are automatically learned from an aligned corpus and (ii) character-level transformation rules, which were manually designed by observing frequent character transformations in the aligned corpus. The ABA normalisation method, which has similarities with the approach of VARD2 developed for English (Baron and Rayson, 2009), works as follows.

Creation of a Word Substitution Lexicon The first step is to learn a word replacement lexicon using a parallel training set. This is done using the classical dynamic programming Needleman-Wunsch alignment algorithm (Needleman and Wunsch, 1970) to optimally align tokenised parallel sentences at the token level, adding a score of 4 for matching words in lowercase (or for *et* and *&* 'and' which are considered equivalent) and a penalty of -1 for word insertions, deletions or mismatches if the non-matching words have a weighted Levenshtein distance of at least 4 or at least the length of each word. For mismatches between words at weighted Levenshtein distance $d < 4$ and strictly smaller than the length of both words, $4 - d$ is the mismatch score taken into account by the alignment algorithm. Note that the weighted Levenshtein distance is computed with a penalty of 1 for insertions and deletions and 2 for character mismatches. These scores were adjusted experimentally after considering the alignment results on a training corpus.

Substitution Step The second step uses this replacement lexicon as well as a contemporary French lexicon built by combining `Morphalou 3.1` (ATILF, 2019) with lexicons of proper nouns developed for CasEN 1.4 (Maurel et al., 2011): `CasEN_Dico.dic`, `Prolex-Unitex-BestOf_2_2_fra.dic` (CasEN Team, 2019) and `Prolex-Unitex_1_2.dic` (Prolex Team, 2013). It proceeds in the following way:

after simple tokenisation[5] of the input text, for each token: 1) if it is present in the contemporary French lexicon, it is kept as it is; 2) otherwise, if it is present in the word replacement lexicon, it is replaced by the associated normalised version in this lexicon; 3) otherwise, it is transformed by a combination of character replacement rules detailed in Appendix A, designed after careful analysis of the aligned words in the training corpus and available in the `apply_rules` function of the `modern.py` script in ABA's distribution:[6] among the obtained candidate tokens, the first one found in the contemporary French lexicon is selected; 4) otherwise, if no candidate generated by character transformation rules is selected, then the original token is kept.

3.2. MT: SMT and NMT

Following promising results for other languages (Scherrer and Erjavec, 2013; Tang et al., 2018) and Modern French (Gabay et al., 2019; Gabay and Barrault, 2020), we provide a comparison of phrase-based statistical MT and NMT.

3.2.1. Phrase-based SMT

The aim of SMT is to automatically find the most probable translation $\hat{t}$ given a source sentence s such that $\hat{t} = \text{argmax}_{t \in T} P(s|t) P(t)$, where $P(s|t)$ models the adequacy of translation, and $P(t)$ the target language model probability, which can be seen as a measure of the fluency/grammaticality of the prediction. The state of the art in SMT is phrase-based MT, where a prediction's score is the sum of scores from various scoring components, including a phrase table (for the translation probability), a language model (for the language model probability), a reordering (or distortion) model and a length penalty. The main implementation used for phrase-based SMT is the Moses toolkit (Koehn et al., 2007), which we use here in this paper.

Phrase-based SMT was the state of the art in MT until around 2015, when NMT first outperformed it (Bahdanau et al., 2015). The main disadvantages of SMT with respect to NMT is the limited ability to model longer distance dependencies and to model semantic relationships between input units, given that probabilities are calculated based on discrete surface forms rather than continuous representations. It nevertheless remains relevant in certain settings, notably when little parallel training data is available (Trieu et al., 2017; Fourrier et al., 2021). For historical spelling normalisation, some works have shown that it can outperform neural approaches, particular in these lower-resource settings (Domingo and Casacuberta, 2018a).

3.2.2. NMT (LSTM and TRANSFORMER)

NMT uses neural networks to find the most probable translation. The standard architecture is an encoder-decoder with an attention mechanism (Bahdanau et al., 2015). The role of the encoder is to encode the source sequence and of the decoder to sequentially produce the target sequence, given the previously translated words and a representation of the input sequence specific to that decoding step (calculated using attention). Importantly, these models work with continuous representations of words, allowing for a greater capacity to generalise across forms and an improved handling of complex linguistic phenomena. The first such models were based on recurrent neural networks (using recurrent units such as LSTM for example), involving sequentially encoding the input and sequentially decoding the output. The current state of the art is the Transformer, which replaces recurrence with self-attention (Vaswani et al., 2017). Transformers have the advantage of speed in training and tend also to perform better, although this does not always hold for very low-resource settings (Fourrier et al., 2021).

NMT model performance is sensitive to the size of the architecture, subword segmentation and training parameters. Sennrich and Zhang (2019) show that previous conclusions about the superiority of SMT systems over NMT in low-resource scenarios do not necessarily hold as long as the NMT parameters are well chosen, highlighting the need to perform adequate parameter search before drawing conclusions. In line with this, we perform extensive hyper-parameter searches of both LSTM and Transformer models (Section 6.3).

3.3. Optional Le*fff*-based post-processing

All three approaches described above rely on parallel training data. Despite the generalisation capabilities of such models, it might be the case that rare situations are not properly dealt with. On the other hand, large-scale lexicons of contemporary French, such as the Le*fff* (Sagot, 2010), can provide high-coverage lexical information regarding the target language of the normalisation process.

Based on this observation, we developed a lexicon-based post-processing tool that can be used after any normalisation model and is based on the Le*fff* (version 3.4). It relies on the idea that a normalised text should mostly contain words known to a large-scale contemporary French lexicon. Any token (whitespace- and/or punctuation-separated character sequence) that does not begin with a capital letter (to avoid proper nouns) and that is unknown to the lexicon is eligible for further normalisation. For every such token, we compute a list of possible normalisations based on a small list of permitted transformations.[7] We then look up all normalisation candidates in our lexicon. If exactly one of the normalisation candidates is known to our lexicon, we replace the input token with this candidate. In all other cases, we leave the token unchanged.

[5] Splitting the sentence on whitespace, the characters . , ! ? ; : and both kinds of apostrophe.

[6] `https://github.com/johnseazer/aba`.

[7] The rules: $Vs \rightarrow \hat{V}$ where V is any vowel, *es* → é, add each possible diacritic to each non-diacritised letter that can have a diacritic (e.g. $u \rightarrow \ddot{u}$), v → u when preceded by a vowel, u → v when preceded by a consonant, $i \rightarrow$ y.

4. Evaluating Normalisation

In terms of automatic metrics, the most commonly used are translation edit rate (TER), word accuracy (based on the gold normalised tokens, non-symmetrised) and some works have used traditional metrics for MT (Gabay and Barrault, 2020), in particular BLEU (Papineni et al., 2002) and CHRF (Popović, 2015). Arguably the most interpretable metric is word accuracy, since it gives an idea about the number of lexical units that would have to be corrected, whereas MT metrics are less interpretable, given that they are designed to incorporate a certain degree of flexibility concerning word order, which is not relevant for the task of spelling normalisation. On the other hand, they have the advantage of penalising predictions that contain additional (hallucinated) tokens as well as correct tokens, a situation that is plausible given the use of sentence-level MT models.

We therefore choose to use a symmetrised version of word accuracy, which is the average between traditional word accuracy (aligning each gold token to predicted (sub)token(s)) and the reverse calculation (aligning each predicted token to gold (sub)token(s)).[8] More details on evaluation can be found in Appendix C. We also evaluate using MT metrics to test how they correlate with word accuracy.

5. Data

For training, development and test data, we present the FREEM corpus (short for *FREnch Early Modern*) called $\text{FREEM}_{\text{norm}}$.[9] The data covers a range of different genres of text throughout different decades of the 17th century, written in prose or verse, which have been semi-automatically normalised (Gabay et al., 2019) and manually corrected. Most of these texts belong to the *belles-lettres* (literature in its broadest sense), which is the type of source we want to normalise, but additional texts from different traditions (science, law, etc.) are present in the corpus. Some of the transcriptions have been produced specifically for this corpus and others have been borrowed from other projects: transcription rules are therefore not strictly equivalent from one text to another regarding, for instance old characters (e.g. *ſ*) or abbreviations (e.g. *õ*→*on*). "Normalisation" is understood here as a partial alignment with contemporary French: in some specific cases, specific spellings are maintained to keep the meter of the verse intact (e.g. the adverbial *-s*: *jusques*+vowel→ *jusques* and not *jusqu'* to maintain the three syllables).

The dataset has been split into train, dev and test sets, for which basic statistics can be found in Table 1. The split was done such that the test set contains a variety of different genres and periods (see Tables 7 and 5 in Appendix B), some of which are covered in the train and dev set and some of which are unseen.

In terms of the difficulty of the task, although many words remain unchanged between the original Modern French and their contemporary French normalisations (75.7% of all words in the training set), there are a non-negligible number of tricky cases. There are a large number of out-of-vocabulary (OOV) items in both the dev and test sets with respect to the training set, and approximately 0.3% of tokens are ambiguous (i.e. they correspond to several possible normalisations depending on the context). Aside minor differences such as punctuation (which is nevertheless not arbitrary, since it can be determined by context), capitals and accents, there are some interesting cases, such as ambiguity concerning verbal conjugations, which may require more contextual information (see Table 2 for two examples). For these cases, it is necessary to normalise words whilst taking into account their context (as in traditional MT). This justifies processing whole sentences rather than isolated words.

6. Experiments

6.1. Baselines

We compare the approaches described in Section 3 with two baseline approaches, the identity function and a basic rule-based approach.

Identity function This keeps the text unchanged.

Rule-based This is a stronger baseline comprising several dozen regular expressions, which were manually written based on simple corpus statistics from our training set. They range from purely typographic rules, which reflect the evolution of the writing system, to lexical rules, which reflect the evolution of the language. Here are a few examples, ordered from purely typographic to fully lexical:

- *ſ* → *s*, *õ* → *om* if followed by *m*, *b* or *p*, or *on* otherwise;
- *i* → *j* at the beginning of a word when followed by a vowel other than *i*;
- *estoit* → *était* and *estoient* → *étaient*.

In addition, we also assess the impact of the lexicon-based post-processing step on these baselines.

6.2. Experimental setup

All NMT models are trained using Fairseq (Ott et al., 2019), with default parameters unless otherwise specified. All models are trained until convergence; the best checkpoint is chosen based on symmetrised word accuracy on the dev set. Subword segmentation is applied using SentencePiece (Kudo and Richardson, 2018) and the BPE strategy (Sennrich et al., 2016).

We train SMT models using Moses (Koehn et al., 2007) and language models using KenLM (Heafield, 2011). We tune using `kbmira` to maximise BLEU.

[8] We first perform character-level alignment using Levenshtein and then realign on the token level with respect to the tokenisation of the gold and predicted sequences respectively.

[9] https://doi.org/10.5281/zenodo.5865428

Set	#sentences.	#tokens ModFr	#tokens Fr	#unique tokens ModFr	#unique tokens Fr	#OOV ModFr	#OOV Fr
Train	17,930	264,311	263,669	21,329	17,238	-	-
Dev	2,443	40435	40294	6736	5993	1,766	1,312
Test	5,706	86,432	86,211	10,457	8,915	3,596	2,530

Table 1: Statistics for the FREEM$_{norm}$ corpus for Modern French (ModFr) and contemporary French (Fr). Texts are tokenised using the Moses tokeniser (Koehn et al., 2007) to calculate statistics and #OOV corresponds to the number of unique out-of-vocabulary tokens.

	Normalisation example 1	Normalisation example 2
nostre 'our'	quel malheur est le **nôtre**	Les larmes sont trop peu pour pleurer **notre** mal
	'what woe is **ours**'	'The tears are too few to cry (for) **our** pain'
appellez 'call'	N'**appelez** point des yeux le Galant à votre aide	...Royaumes, par nous vulgairement **appelés** Siam
	'Do not **call** the Galant for help with your eyes'	'...kingdoms, **known** popularly by us as Siam'

Table 2: Two examples of context-dependent ambiguity (Modern French words *nostre* and *appellez*) when normalising to contemporary French.

6.3. Best Model Search

6.3.1. Neural models

For LSTM and Transformer models, we performed hyper-parameter searches to maximise the symmetrised word accuracy on the development set. We explored (i) the network size (cf. Table 3 for LSTM models and Table 4 for Transformer models), (ii) the degree of subword segmentation via different BPE vocabulary sizes (500 1k, 2k, 4k, 8k, 16k, 24k), (iii) the learning rate (0.0005, 0.001, 0.001) and (iv) the batch size (1000, 2000, 3000, 4000 tokens). In order to avoid having to explore the combination of all parameters, we explored hyper-parameters in a step-wise fashion from (i) to (iv), keeping the best parameters from the previous step. We then explored variations on the network size parameters, varying attributes one below and one above the default values. Results were calculated as an average of three differently seeded runs for each combination. We began with default values for all hyper-parameters and varied only those mentioned.

Both models performed best with a BPE vocabulary of 1k, batch size of 3000 and learning rate of 0.001. The best network sizes were M for the LSTM, and a variant of the M model for the Transformer, with only 2 encoder layers rather than 4.

6.3.2. Statistical MT model

As for the neural models, we test several different granularities of segmentation: character-based, 500, 1k and 2k.[10] We use a 4-gram language model trained on the target side of either the parallel training data or the normalised texts of the FREEM$_{max}$ corpus (Gabay et al., 2022). The best subword segmentation is with vocabulary size 500 (interestingly not character-based as what has previously been used) and with the language model trained on the target side of the parallel training data.

[10]Larger vocabulary sizes result in worse scores and were also more difficult to train because of memory problems.

Size	#enc. layers	#dec. layers	embed. dim.
XS	1	1	128
S	2	2	256
M	3	3	384
L	4	4	512

Table 3: Network sizes explored for LSTM models.

Size	#attn. heads	#layers enc.	#layers dec.	Dim. embed.	Dim. ffwd.
S	2	2	2	128	512
M	4	4	4	256	1024
L	8	6	6	512	2048

Table 4: Network sizes explored for Transformer models. L corresponds to Transformer-base.

7. Results

We compare the approaches described in Section 3 according to the three evaluation metrics discussed in Section 4: symmetrised word accuracy (written as WordAcc), BLEU and CHRF.

Results are shown in Table 5. For MT approaches, we run each model three times with three random seeds and report the average score and standard deviation. Models (1)-(4) are baselines and already achieve relatively high scores. This is unsurprising, given the large number of words that do not need modifying: the identity function (copying the source text) gives 72.73% word accuracy. The rule-based approach is significantly better than the first baseline, and adding the post-processing step (+L*ffff*) considerably improves both results. The two statistical approaches, the hybrid ABA and SMT, both perform better than the baselines, with SMT actually performing the best out of all approaches. The NMT models perform slightly worse according to all metrics than SMT. Although the scores

	Model	WordAcc (%)	BLEU	ChrF	OOV WordAcc (%)
Baseline models					
(1)	Identity	72.73	40.25	73.77	43.00
(2)	Identify + Le*fff*	86.12	66.78	87.40	64.84
(3)	Rule-based	89.05	72.47	89.94	60.22
(4)	Rule-based + Le*fff*	90.85	76.90	91.70	66.51
Alignment-based approach					
(5)	ABA	95.14	87.70	95.84	69.50
MT approaches					
(6)	SMT	**97.10**±0.02	**92.59**±0.05	**97.71**±0.01	75.64±0.18
(7)	LSTM	96.14±0.08	91.77±0.21	96.85±0.08	**76.69**±0.70
(8)	TRANSFORMER	95.89±0.07	91.30±0.08	96.65±0.05	75.73±0.38
+ Lexicon-based post-processing					
(9)	ABA + Le*fff*	95.44	88.37	96.13	73.54
(10)	SMT + Le*fff*	**97.24**±0.02	**92.97**±0.05	**97.85**±0.01	**78.37**±0.20
(11)	LSTM + Le*fff*	96.25±0.10	92.07±0.25	96.95±0.10	**78.35**±0.79
(12)	TRANSFORMER + Le*fff*	96.01±0.09	91.62±0.14	96.76±0.08	77.51±1.00

Table 5: Results on the test set. "+ Le*fff*" indicates that the lexicon-based post-processing was applied.

Table 6: Word accuracy on OOV target tokens (%)

of LSTM and TRANSFORMER are very similar, LSTM scores are slightly higher. It is an interesting finding that the SMT outperforms NMT in our scenario, as this goes against recent findings for Modern French (Gabay and Barrault, 2020), despite us having more parallel data available. As for the baselines, adding the post-processing step improves both statistical and neural models, with the best result being SMT+Le*fff* with a symmetrised word accuracy of 97.24%.

As recommended by Robertson and Goldwater (2018), we also calculate word accuracy for OOV tokens (based on the gold tokens). Results (Table 6) show that the highest scoring model for OOV accuracy is LSTM, although if post-processing is applied, both SMT and LSTM show similar scores. Adding the post-processing step significantly helps the OOV accuracy of all methods, showing that it is an important complementary step.

The three evaluation metrics reveal the same pattern in results for these models, with BLEU varying more in absolute scores than the other metrics.

8. Comparative Analysis

8.1. How Similar are the Methods?

In Figure 2, we compare the predictions token by token and report the percentage of identical normalisations between methods.[11] Unsurprisingly, the neural methods (LSTM and TRANSFORMER) are most similar to each other. SMT is the most similar to TRANSFORMER and ABA is most similar to SMT.

8.2. Conservative or Zealous?

Depending on how the tool is to be applied, it can be better to have a more conservative or zealous model.

[11]The analysis is computed against the first prediction for methods for which three random seeds were used.

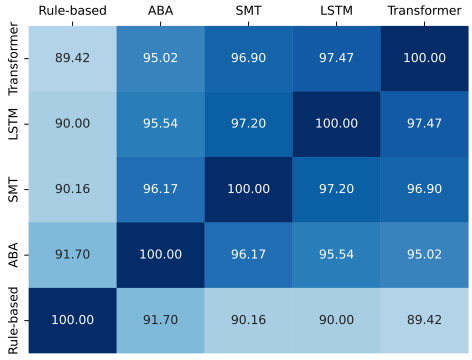


Figure 2: The percentage of identically normalised test set tokens between methods.

If automatic normalisation is to be used as a pre-annotation tool to help experts manually normalise texts, it is important for the automatic step not to introduce serious errors that could be more difficult to detect and time-consuming to correct. This is a concern notably for NMT-based models (Gabay and Barrault, 2020), which can be more creative in their transformation than either rule-based or SMT-based approaches. It may however be less of a problem if normalisation is to be used for certain downstream tasks using standard contemporary NLP tools (e.g. PoS-tagging or parsing). This is because a more zealous normalisation could provide better performance (by providing contemporary word forms), without the word forms themselves having to necessarily correspond to the correct ones.

To compare the methods for their conservativeness/zealousness, we align the output of each method with the source text and calculate (i) how often it changes a token that should have been kept as it is (Table 3), and (ii) how often it leaves a token untouched

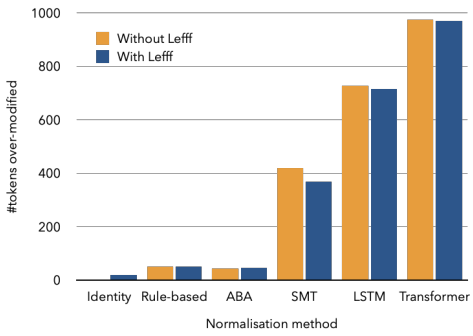


Figure 3: Comparison in the number of 'over-modified' test set tokens for each method.

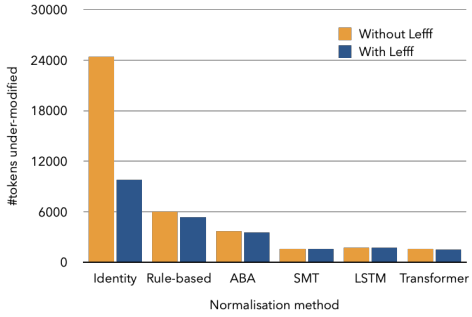


Figure 4: Comparison in the number of 'under-modified' test set tokens for each method.

that should be modified (Table 4). The identity function, rule-based system and ABA rarely over-modify, contrarily to SMT and NMT. Logically, the methods show the the inverse pattern for under-modification, with the identity and rule-based approaches being the most conservative and under-modifying the most. The SMT and NMT models under-modify at very similar rates, suggesting that performance differences could largely stem from over-modification rather than how much they under-modify. The best method, SMT, has the lowest rate of under-modification and a medium level of over-modification. ABA is interesting, because it under-modifies less than the baselines and yet does not over-modify as much as the MT approaches.

Adding the Le*fff*-based post-processing step has the effect of both correcting some over-modifications that were introduced and providing normalisations for previously unmodified tokens, thereby significantly improving the processing of OOV words.

8.3. Qualitative analysis of approaches

In this section, we compare the results of the best rule-based approach, ABA + Le*fff* and the best MT approach, SMT + Le*fff*, by using an alignment of the normalised versions of the dev data (available at `https://freem-corpora.github.io/models/norm_model/`).

Unsurprisingly, given that the substitution rules are not contextual, ABA + Le*fff* makes many errors in ambiguous cases, such as *A* instead of *À*, *prés* instead of *près*, *voila* instead of *voilà*, or *mes feux redoublez* instead of *mes feux redoublés*. Taking into account frequency scores either for the word replacement or for the character transformation rules in the training data may help avoid those mistakes. ABA + Le*fff* is also very sensitive to mistakes in the training corpus. For example, it succeeds in transforming *auoient* into *avaient* but not *avoient*, whereas SMT + Le*fff* succeeds. It also lacks some rules. For example it has no rule to normalise double consonants (for example *principalles* normalised into *principales*, *assouppit* into *assoupit*), whereas SMT + Le*fff* performs pretty well in this case. The SMT approach displays some more creative errors, but which appear easy spot if the normalised text is manually proof-read), e.g. *ma pẽſée* transformed into *ma pmentsée*. It is also prone to deleting certain words such as determiners, possibly because in some contexts they are less probable according to the language model. Finally, considering the fact that it is often the case that, when one of the two methods makes a mistake, the other one performs a correct normalisation, finding a relevant post-processing approach seems like a promising way to increase the quality of the results.

9. Conclusion

We have presented FREEM$_{\text{norm}}$, a new benchmark for the normalisation of Early Modern French, and compared a range of normalisation methods, including an alignment-based approach and various MT-based methods, with SMT outperforming all other approaches. Adding a post-processing with a contemporary French lexicon systematically helps, particularly for OOV tokens. We compare the strengths of the different methods, with rule- and alignment-based approaches being more conservative and MT approaches being less so. While MT approaches achieve the best accuracy, a model such as the alignment-based ABA is possibly more adapted to pre-annotation as it offers a good compromise between making good normalisation choices without overly normalising tokens that should not have been modified. We release all our data, models and scripts to encourage further research on this topic by the digital humanities community.

Acknowledgements

This work was performed using HPC resources from GENCI-IDRIS (Grant 20-AD011012254). It benefits from a State funding managed by the National Research Agency (ANR) under the Investments for the Future program (reference ANR-16-IDEX-0003, I-Site FUTURE, *Cité des dames, créatrices dans la cité*) in addition to the contributions of institutions and partners involved. It was also partly funded by the first and penultimate authors' chairs in the PRAIRIE institute funded by the French national agency ANR as part of the "Investissements d'avenir" programme under the reference ANR-19-P3IA-0001.

10. Bibliographical References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA.

Baron, A. and Rayson, P. (2009). Automatic standardisation of texts containing spelling variation: How much training data do you need? In *Proceedings of the Corpus Linguistics Conference: CL2009*, University of Liverpool, UK.

Bollmann, M. and Søgaard, A. (2016). Improving historical spelling normalization with bi-directional LSTMs and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, pages 131–139, Osaka, Japan.

Bollmann, M., Petran, F., and Dipper, S. (2011). Rule-based normalization of historical texts. In *Proceedings of the Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, pages 34–42, Hissar, Bulgaria.

Bollmann, M. (2012). (Semi-)automatic normalization of historical texts using distance measures and the Norma tool. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*, pages 3–14, Lisbon, Portugal.

Bollmann, M. (2019). A large-scale comparison of historical text normalization systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3885–3898, Minneapolis, Minnesota.

Domingo, M. and Casacuberta, F. (2018a). A Machine Translation Approach for Modernizing Historical Documents Using Back Translation. In *Proceedings of the 15th International Workshop on Spoken Language Translation (IWSLT 2018)*, pages 38–47, Bruges, Belgium.

Domingo, M. and Casacuberta, F. (2018b). Spelling Normalization of Historical Documents by Using a Machine Translation Approach. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 129–137, Alicante, Spain.

Domingo, M. and Casacuberta, F. (2021). A comparison of character-based neural machine translations techniques applied to spelling normalization. In Alberto Del Bimbo, et al., editors, *Pattern Recognition. ICPR International Workshops and Challenges*, pages 326–338, Cham. Springer International Publishing.

Domingo, M., Chinea-Rios, M., and Casacuberta, F. (2017). Historical documents modernization. *The Prague Bulletin of Mathematical Linguistics*, 108:295–306.

Fix, H., (1980). *Automatische Normalisierung - Vorarbeit zur Lemmatisierung eines diplomatischen altisländischen Textes*, pages 92–100. Max Niemeyer Verlag.

Fourrier, C., Bawden, R., and Sagot, B. (2021). Can cognate prediction be modelled as a low-resource machine translation task? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 847–861, Online.

Gabay, S. and Barrault, L. (2020). Traduction automatique pour la normalisation du français du XVIIe siècle. In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 2 : Traitement Automatique des Langues Naturelles*, pages 213–222, Nancy, France.

Gabay, S., Riguet, M., and Barrault, L. (2019). A Workflow For On The Fly Normalisation Of 17th c. French. In *Proceedings of the 2019 Digital Humanities Conference*, Utrecht, Netherlands.

Gabay, S. (2021). Beyond Idiolectometry? On Racine's Stylometric Signature. In Maud Ehrmann, et al., editors, *Conference on Computational Humanities Research 2021*, pages 359–376, Amsterdam, Netherlands.

Hämäläinen, M., Säily, T., Rueter, J., Tiedemann, J., and Mäkelä, E. (2018). Normalizing early English letters to present-day English spelling. In *Proceedings of the Second Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 87–96, Santa Fe, New Mexico.

Hauser, A. W. and Schulz, K. U. (2007). Unsupervised Learning of Edit Distance Weights for Retrieving Historical Spelling Variations. In *Proceedings of the First Workshop on Finite-State Techniques and Approximate Search*, pages 1—6.

Heafield, K. (2011). KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Kogkitsidou, E. and Gambette, P. (2020). Normalisation of 16th and 17th century texts in French and geographical named entity recognition. In *Proceedings of the 4th ACM SIGSPATIAL Workshop on Geospatial Humanities*, pages 28–34, Seattle, Washington,

USA.

Korchagina, N. (2017). Normalizing medieval German texts: from rules to deep learning. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 12–17, Gothenburg. Linköping University Electronic Press.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium.

Ljubesic, N., Zupan, K., Fiser, D., and Erjavec, T. (2016). Normalising Slovene data: historical texts vs. user-generated content. In Stefanie Dipper, et al., editors, *Proceedings of the 13th Conference on Natural Language Processing, KONVENS 2016*, volume 16 of *Bochumer Linguistische Arbeitsberichte*, pages 146–155, Bochum, Germany.

Maurel, D., Friburger, N., Antoine, J.-Y., Eshkol, I., and Nouvel, D. (2011). Cascades de transducteurs autour de la reconnaissance des entités nommées. *Traitement automatique des langues*, 52(1):69–96.

Mitankin, P., Gerdjikov, S., and Mihov, S. (2014). An approach to unsupervised historical text normalisation. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, pages 29–34, Madrid, Spain.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, USA.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Pettersson, E., Megyesi, B., and Nivre, J. (2013a). Normalisation of historical text using context-sensitive weighted Levenshtein distance and compound splitting. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NoDaLiDa 2013)*, pages 163–179, Oslo, Norway.

Pettersson, E., Megyesi, B., and Tiedemann, J. (2013b). An SMT Approach to Automatic Annotation of Historical Text. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NoDaLiDa 2013)*, pages 54–69, Oslo, Norway.

Pettersson, E., Megyesi, B., and Nivre, J. (2014). A Multilingual Evaluation of Three Spelling Normalisation Methods for Historical Text. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 32–41, Gothenburg, Sweden.

Piotrowski, M. (2012). *Natural language processing for historical texts*, volume 5(2) of *Synthesis lectures on human language technologies*. Morgan & Claypool Publishers.

Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal.

Porta, J., Sancho, J.-L., and Gomez, J. (2013). Edit Transducers for Spelling Variation in Old Spanish. In *Proceedings of the Workshop on Computational Historical Linguistics (NoDaLiDa 2013)*, pages 70–79, Oslo, Norway.

Reynaert, M., Hendrickx, I., and Marquilhas, R. (2012). Historical spelling normalization. A comparison of two statistical methods: TICCL and VARD2. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*, Lisbon, Portugal.

Robertson, A. and Goldwater, S. (2018). Evaluating historical text normalization systems: How well do they generalize? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 720–725, New Orleans, Louisiana.

Scherrer, Y. and Erjavec, T. (2013). Modernizing historical Slovene words with character-based SMT. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, pages 58–62, Sofia, Bulgaria.

Sennrich, R. and Zhang, B. (2019). Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

Tang, G., Cap, F., Pettersson, E., and Nivre, J. (2018). An Evaluation of Neural Machine Translation Models on Historical Spelling Normalization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1320–1331, Santa Fe, New Mexico, USA.

Tjong Kim Sang, E., Bollmann, M., Boschker, R., Casacuberta, F., Dietz, F., Dipper, S., Domingo, M., van der Goot, R., van Koppen, M., Ljubešić, N., Östling, R., Petran, F., Pettersson, E., Scherrer, Y., Schraagen, M., Sevens, L., Tiedemann, J., Vanalle-

meersch, T., and Zervanou, K. (2017). The CLIN27 shared task: Translating historical text to contemporary language for improving automatic linguistic annotation. *Computational Linguistics in the Netherlands Journal*, 7:53–64.

Trieu, H. L., Tran, D.-V., and Le Nguyen, M. (2017). Investigating phrase-based and neural-based machine translation on low-resource settings. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pages 384–391, Cebu City, Philippines.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. U., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Vilar, D., Peter, J.-T., and Ney, H. (2007). Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39, Prague, Czech Republic.

11. Language Resource References

ATILF. (2019). Morphalou. `https://hdl.handle.net/11403/morphalou/v3.1`. ORTOLANG (Open Resources and TOols for LANGuage).

CasEN Team. (2019). Casen 1.4. `https://tln.lifat.univ-tours.fr/medias/fichier/casen-fr-1-4_1596032302677-zip?ID_FICHE=332027&INLINE=FALSE`.

Dipper, S. and Schultz-Balluff, S. (2013). The Anselm Corpus: Methods and perspectives of a parallel aligned corpus. In *Proceedings of the workshop on computational historical linguistics at NoDaLiDa 2013*, pages 27–42, Oslo, Norway.

Erjavec, T., Ringlstetter, C., Žorga, M., and Gotscharek, A. (2011). A lexicon for processing archaic language: the case of XIXth century Slovene. In *First International Workshop on Lexical Resources*.

Gabay, S., Bartz, A., Chagué, A., and Gambette, P. (2022). FreEM max. `https://github.com/FreEM-corpora/FreEMmax_OA`.

Ortiz Suarez, P., Gabay, S., Bartz, A., Bawden, R., Sagot, B., and Gambette, P. (2022). From FreEM to D'AlemBERT: a Large Corpus and a Language Model for Early Modern French. In *Proceedings of the 13th International Conference on Language Resources and Evaluation (LREC'22)*, Marseille, France.

Prolex Team. (2013). Prolex 1.2. `https://tln.lifat.univ-tours.fr/medias/fichier/prolex-unitex-1-2_1562935068094-zip?ID_FICHE=321994&INLINE=FALSE`.

Riguet, M. (2019). Normalisa, Script à base de règles pour normaliser les textes français du XVI^{e} au XIX^{e} siècle. `https://github.com/mriguet/Normalisa/`.

Sagot, B. (2010). The Le*fff*, a Freely Available and Large-coverage Morphological and Syntactic Lexicon for French. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

The French Wikisource Community. (2022). Wikisource:dictionnaire. `https://fr.wikisource.org/wiki/Wikisource:Dictionnaire`.

A. ABA Normalisation Rules

The character transformation rules used in the second step of ABA include → s, ß → ss, & → et; the resolution of letters with a tilde used to abbreviate an n or an m; sç → s; final oing → oin; final y → i; sch → ch; aye → aie, oye → oie. The obtained word is considered as an initial candidate followed by the supplementary candidates obtained with the following rules: ct → t; vowel followed by dv → same vowel followed by v; final ans → ands, final ens → ends, final ans → ants, final ens → ents; final ois → ais (same with oit and oient); final ez → és, final és → ez; st → t, est → ét; as followed by m n q or t → â followed by the same letter (same with es, is, os and us); y → i; ü or eü → u. Finally, for all generated candidates, the following transformation rules are applied: is → î, ai → aî, u → v, v → u, non final e not followed by s → é.

B. Distribution of the Datasets by Decade and Genre

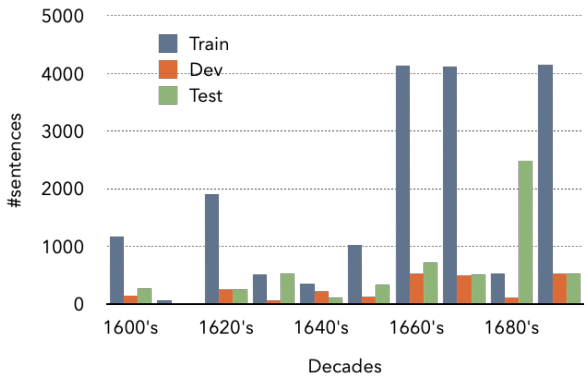


Figure 5: Distributions of data in terms of decades.

Genre	Train	Dev	Test
Caractères	190	25	25
Comédie	4870	619	623
Tale	120	15	15
Correspondence	1533	198	199
Law	61	0	0
Fables	899	112	114
Journalism	142	0	0
Medicine	0	59	114
Philosophy	455	57	200
Physics	0	0	182
Poetry	1777	224	226
Novel	1071	132	730
Memoir novel	213	27	27
Theology	560	70	72
Tragedy	5847	708	3155
Travel	192	24	24

Table 7: Number of sentences per genre.

C. Evaluation details

Word accuracy is calculated by aligning the set of sentences (each reference sentences and its normalised sentence) on the character level and then using the alignment matrix to produce a token-level alignment.

Initial Character-level Alignment Character-level alignment is performed using a modified (weighted) version of Levenshtein, whereby certain characters are considered equivalent (e.g. accented and non-accented versions of characters, long s ( ) and s). The alignment is also designed to avoid tokenisation and punctuation mismatches unless they are really necessary for a successful alignment:

- by default, the cost of a substitution is 1, whereas the cost of an insertion or a deletion is 0.8;
- the cost of a substitution of a reference white-space character with a non-white-space is prohibitive (1,000,000);
- the cost of a substitution of a reference non-white-space character with a white-space is 30;
- the cost of a substitution involving a punctuation mark (within ,.;-!?') is 20;
- the cost of the deletion of a white-space character in the reference is prohibitive;
- the cost of the insertion of a white-space character in the reference is 2.

Token-level alignment The token-level alignment must necessarily be carried out with respect to the tokenisation of one of the sequences (there is not always a one-to-one mapping between reference and normalised tokens). We carry out tokenisation prior to character-level alignment using a very basic tokeniser lightly adapted to French (breaking on whitespace and around punctuation) and use then use whitespace tokens to delimit tokens when token-aligning the two sequences. We can either take the tokenisation of the reference sequence or of the normalised sequence as the basis for alignment. We preserve information about token boundaries such that different segmentations will be penalised even if the non-whitespace characters are identical.

(1) Ref: surtout j'ai choisi davantage ses écrits
MT: sur tout ji choisi d'avantage ses escrits,

(2) Align: surtout|||sur␣␣␣␣tout j'|||j
ai|||i choisi davantage|||d'␣␣avantage ses écrits|||escrits

For example, given a reference (Ref) and a predicted normalisation (MT) as shown in Example 1, the alignment in Example 2 is produced, where:

- ||| indicates that the reference and MT output do not match for that token;

- ▁ indicates that there is a token boundary introduced by the tokeniser in the aligned sequence of characters. Where there is also a space in the original sequence (before tokenisation), a double ▁▁ is indicated (case of over-merging);
- ▒ indicates that there is no token boundary to the right (case of over-splitting).

Symmetrised Accuracy Once aligned, the accuracy is the number of tokens for which the corresponding token is identical divided by the total number of tokens. We calculate a symmetrised accuracy, which is the average between the two accuracies: (i) the reference sentences are used as the basis for alignment and (ii) the normalised sentences are used as the basis for alignment. This is important because it helps to penalise very poor normalisations, such as those that can be produced by some MT-style models, where words can be hallucinated. If the accuracy is only computed according to the reference tokenisation, it is possible for all hallucinated words to be aligned to a single reference token and therefore penalised very little with respect to the amount of noise added.

The Evolution of the Idiolect over the Lifetime: A Quantitative and Qualitative Study of French 19th Century Literature

Olga Seminck, Philippe Gambette, Dominique Legallois, Thierry Poibeau

CA Journal of Cultural Analytics

Seminck, Olga, et al. "The Evolution of the Idiolect over the Lifetime: A Quantitative and Qualitative Study of French 19th Century Literature." *Journal of Cultural Analytics*, vol. 7, no. 3, Sept. 2022, doi:10.22148/001c.37588.

ARTICLE

The Evolution of the Idiolect over the Lifetime: A Quantitative and Qualitative Study of French 19th Century Literature

Olga Seminck[1], Philippe Gambette[2], Dominique Legallois[1], Thierry Poibeau[1]

[1] Laboratoire Langues, Textes, Traitements informatiques, Cognition UMR 8094, [2] Laboratoire d'Informatique Gaspard-Monge UMR 8049

Keywords: idiolect, French literature, authorship, stylometry, literature

https://doi.org/10.22148/001c.37588

Journal of Cultural Analytics

Vol. 7, Issue 3, 2022

The way in which authors express themselves is unique but changes over their lifetime. However, quantitative studies of this idiolectal evolution are rare. Using the Corpus for Idiolectal Research (CIDRE) that contains the dated works of 11 prolific 19th century French fiction writers, we propose new methods to identify, quantify and describe the grammatical-stylistic changes that take place using lexico-morphosyntactic patterns, also called motifs. To examine the strength of the chronological signal of change, we developed a method to calculate if a distance matrix of literary works contains a stronger chronological signal than expected by chance. Ten out of 11 corpora showed a higher than chance chronological signal, leading us to conclude that the evolution of the idiolect is in a mathematical sense monotonic, supporting the rectilinearity hypothesis previously put forward in the stylometric literature. The rectilinear property of the evolution of the idiolect found for most authors in CIDRE subsequently enabled us to propose a machine learning task: predicting the year in which a work was written. For the majority of the authors in our corpus, the accuracy and the amount of variance that is explained by the model were high and we discuss why the technique might fail for others. After applying a feature selection algorithm, we examined the most important features, i.e. the motifs that have the greatest influence on idiolectal evolution. We find that some of those features are stylistic and have been previously identified in qualitative literature studies. We report some remarkable stylistic constructions revealed by our algorithm to illustrate which kind of stylistic patterns can be extracted using our method.

1. Introduction

Is it true that we do not speak at 20 as we do at 60? In this article we examine if an individual's representation of a language — the idiolect — and the utterances that are its product, are fixed once and for all or not. Little research has been carried out to characterize and measure the evolution of the idiolect in an extensive manner. Our main goal was thus to develop methods in this direction that can also be applied to other longitudinal corpora. We evaluated our methods on a corpus containing the idiolects of 11 French fiction writers.

There are several reasons why it is interesting to take into account interpersonal variation over time. First, the notion of idiolect is relevant for corpus linguistics. As Heck stressed, idiolects are the primary objects of study in linguistics (in the end, we can only observe utterances that are the products of idiolects). However, most often corpora are considered as homogenous, and do not take into account the influence of individual authors on their content. It is somehow assumed that the large number of different authors in a resource will erase any individual differences. But taking these differences into account could

help better understand to what extent some features are specific to a genre, a community or, on the contrary, to an individual author or speaker. In this article, we focus on the chronological evolution of the idiolect.

Let us start by introducing some important terminology. The first term that needs to be clarified is idiolect. In Bloch's original definition, the idiolect represents: "The totality of the possible utterances of one speaker at one time in using a language to interact with one other speaker." More recently, Dittmar (111) proposed this definition: an idiolect is "the language of the individual, which because of the acquired habits and the stylistic features of the personality differs from that of other individuals and in different life phases shows, as a rule, different or differently weighted CM [communicative means]".

In addition to this, we believe that the definition of idiolect should take into account the fact that every utterance (written or oral) of an individual is part of a particular discursive practice — or, put differently, of a particular textual genre (informal conversation, tweet, philosophical essay, etc.). The idiolect should thus necessarily be considered in relation to a particular practice: it corresponds to the use by an individual of only part of the possible linguistic forms related to a discursive practice. Bloch takes this into account when he states "that a given speaker may have different idiolects at successive stages of his career, and [...] that he may have two or more different idiolects at the same time."

Another relevant notion is style. Style corresponds to linguistic forms that an observer considers as remarkable from an aesthetic point of view, in a particular discourse, compared to the discourse of others. Although the definitions of idiolect and style are intertwined, especially for those working with literary corpora, the main difference is that, for stylistic studies, a judgement has generally to be performed on the stylistic value of the linguistic phenomena under study. It should be noted that the notion of stylistic judgement is in itself highly subjective, and no clear criteria seem to be available to determine what has a stylistic value and what does not. Consequently, we focused on the evolution of idiolects, instead of styles, so as to avoid aesthetic judgements. We therefore use Bloch's definition of the idiolect that includes "the totality of possible utterances", instead of Dittmar's that focuses on stylistic features.

In this article, after the presentation of related work, we present two computational experiments to study the chronological evolution of the idiolect, followed by a qualitative analysis. We start by examining the rectilinearity hypothesis mentioned in the work of Stamou, i.e. *"the hypothesis that certain aspects of an author's writing style evolve rectilinearly over the course of an author's lifetime, hence with appropriate methods and stylistic markers, such changes ought to be detectable"*. First, we evaluate the chronological signal in corpora including the dated works of French 19th century authors by so-called Robinsonian matrices. Second, we build linear regression models for

each author studied to see whether it is possible to predict the year in which a particular novel was written by extrapolation from other works by the same author. Our regression models rely on special linguistic-stylistic patterns, called *motifs* (Legallois et al.) and are able to identify the patterns that play the greatest role in the chronological evolution of the idiolect. We discuss the stylistic value of some of these motifs in a qualitative analysis presented in section 6. The article ends with a discussion and conclusion.

2. Literature Review

Our work is directly in line with the notion of stylochronometry, a special research "niche" that studies the diachronic evolution of style. The term was coined by Forsyth and encompasses the characterization of style according to different time periods, as well as the attribution of tentative dates to literary works. Stamou reviewed a large number of studies on this topic, discussing literary works by writers such as the poet W.B. Yeats (Jaynes), or the prose of Samuel Becket (Opas), to the lyrics of the Beatles (Whissell). The review draws some important conclusions that are still valid today. The first is that even though dating methods would eventually be most useful to date works with an uncertain date of creation — such as texts by Plato, Euripides and Shakespeare — these methods should be developed, tested and evaluated using gold standard corpora (where texts can be reasonably associated with a precise date of creation so as to get a solid performance evaluation), a criterion that was already underlined by Forsyth. Despite this, there is a large literature on the topic of stylochronometry with experiments investigating only works with problematic dating (e.g. Cox and Brandwood; Wishart and Leach; T. M. Robinson; Ledger; Temple on Plato's works; Devine and Stephens; Cropp and Fick; Smith and Kelly on Euripides, and Brainerd; Derks; Jackson on Shakespeare), making it difficult to evaluate the results. Experiments in which the methods used are carefully compared to reference corpora with known dates are rather rare (however, see Can and Patton, on two modern Turkish writers). Daelemans also underlines this evaluation problem and suggests using evaluation methods from the field of Natural Language Processing (NLP). In the same vein, Craig states that "stylistic analysis needs finally to pass the same tests of rigor, repeatability, and impartiality as authorship analysis if it is to offer new knowledge".

Specifically for works on French, we came across studies using off-the-shelf methods developed for statistical textual analysis (Pincemin), for instance stylo R (Eder et al.), Lexico (Lamalle et al.), TXM (Heiden et al.), Le Trameur (Fleury and Zimina), and Hyperdeep (Vanni et al.). For example, Guaresi et al. study the evolution of style on a corpus of the annals of the congress of the French communist party from 1936 to 2018 using correspondence analysis on the vocabulary. However, a drawback of off-the-shelf methods is that they provide mainly exploratory analyses or visualizations which leave considerable room for interpretation and cannot be used directly to focus on specific stylochronometry questions or hypotheses with a rigorous evaluation

procedure. We will therefore not go into detail about this type of study but will instead discuss more focused studies that have, in our opinion, developed interesting and relevant approaches for the task.

Mollin investigated Tony Blair's idiolect. Her goal was to identify "maximizer collocations" (collocations involving adverbs such as *fully*, *entirely* or *absolutely*) that were specific to Blair's idiolect, when compared to the English language in general (comparing a three million word corpus of Tony Blair with the British National Corpus (BNC XML Edition)). Collocations were selected using the three measures: relative frequency, Mutual Information (Church et al.) and the log-likelihood measure (Dunning). A series of collocations that are typical of Blair's idiolect was identified using these three measures.

Mollin's article nicely combines quantitative and qualitative analysis and presents a clear methodology. Unfortunately, we could not find an online accessible repository of the Tony Blair Corpus, but the methods are explained to an extent that the research should be replicable. Mollin's results suggest that the notion of idiolect is indeed a relevant linguistic concept, and that there are some linguistic patterns that are highly idiosyncratic for a speaker (even though her study only included one individual).

A second researcher working on the notion of idiolect is Barlow. He studied the idiolect of five White House Press Secretaries who held this function from 1 to 4 years. For each person the author collected a corpus of approximately 200K to 1200K tokens and compared the individual frequencies of the most frequent bigrams (lexical and Part of Speech) of each press secretary against the others. He showed that individual patterns are highly recognizable and that inter-speaker variability is much larger than intra-speaker variability. Moreover, he found that the inter-speaker differences were "core aspects of language and not peripheral idiosyncrasies", meaning that they play a role in the use of function words and high frequency words, such as 'by the' and 'we have'. He also found that the speech of an individual remained remarkably stable over time, but of course, one needs to keep in mind that the maximum period for a secretary in the corpus was only four years.

Another study that concluded in favour of the staticness of the idiolect is Meyerhoff and Walker. They tried to determine to what extent the grammar of individuals is morpho-syntactically similar to that of a community. They studied the absence of the verb 'be' in a community speaking an English-based creole compared to other members of this community who had joined an urban community speaking a more 'standard' English. Their conclusions are mixed, suggesting that, despite the possibility of the idiolect evolving, conservatism can also play an important role. However, it should be noted that this study only applied to one grammatical construction in a multilingual setting, so that the reported results may be hard to generalize.

Evans wrote her PhD thesis on diachronic morpho-syntactic changes in the idiolect of Queen Elizabeth I from a sociolinguistic perspective by comparing her letters, speeches and translations (forming a corpus of 78K tokens) from before her ascension to those from the period after this event, which is often speculated to have had the greatest influence on Elizabeth's language by other scholars. Interestingly, Evans used a reference corpus — the Corpus of Early English Correspondence (Raumolin-Brunberg and Nevalainen) — and previous studies on it to identify 9 morpho-syntactic features present in this corpus and the corpus of Queen Elizabeth. The goal was to see whether the two corpora evolved in the same way. The author found that the ascension to the throne only influenced two features (the increase in the use of the royal 'we' and the decrease in the use of periphrastic superlative adjectives), and that time in general and the long education of Queen Elizabeth had a constant influence on the development of her idiolect. Evans' study provides a good example of how corpus linguistics and qualitative analysis can jointly contribute to the study of the evolution of the idiolect.

In the field of stylochronometry, the work of Klaussner and Vogel of 2015 and 2018 and Klaussner of 2017 should be mentioned. They developed regression models and evaluated them on two individual corpora of North American writers, a reference corpus and against a baseline. They used a machine learning task that aimed to predict the year of writing of a given work, using a relevant evaluation metric. Their methods play an important role in the second part of our quantitative study (Section 5) and will therefore be discussed in more detail below. However, we can already conclude that the methods proposed by Klaussner and Vogel show how quantitative machine learning methods can be used to fuel qualitative research on stylistic changes.

We have said that relevant large scale resources in this domain are scarce. We should however mention a large recent resource: the EMMA corpus (Petré et al.). It features 87 million words of prolific English 17th century writers. Various studies on the evolution of the idiolect were conducted using this resource, for example Petré and Van de Velde — although using an earlier slightly smaller version of the corpus — investigated the role of individual language users and the language community in the semantic and morphosyntactic process of grammaticalization of a specific construction: 'be going to INF'. They show how the rate of this construction was influenced before, during and after its conventionalization and observe differences between generations. Their method shows how the process of grammaticalization can be studied for individuals but also for a community at the same time. Anthonissen and Petré also show how the use of larger corpora helps to study lifespan changes affecting syntactic constructions; they demonstrate by the example of the construction 'be going to' that individual writers in the EMMA corpus can adopt and continue to participate in grammatical innovations during adulthood.

Contrary to most of the studies examined in this section, our approach takes into account all kinds of patterns (called motifs) and not only a handful of predefined and carefully selected sequences. With this more comprehensive approach, we hope to produce a more robust and reliable model of the evolution of the idiolect.

3. Corpus

For this study, we used the Corpus for Idiolectal Research (CIDRE) (Seminck et al.). This corpus features 11 French 19th and early 20th century prolific fiction writers, with a total of 37 million words. All the works have been carefully dated and the corpus includes only works of fiction, so that the problem of individuals having different idiolects related to their social situation does not interfere.

To address the question of diachronic language evolution in general in opposition to idiolectal evolution, we assembled a 'reference corpus'. This corpus contains 361 works of fiction by French authors from the same time period as the works in CIDRE, but no particular attention was paid to individual authors: they can be included in the resource if they wrote only one work. To assemble the reference corpus, we used the online tool GutenTag (Brooke et al.) that enables one to download a subcorpus from Project Gutenberg. With a semi-automatic approach to run GutenTag several times, to filter sufficiently long books and to automatically date the first edition of each work using the catalogue of the Bibliothèque nationale de France, we were able to obtain a total of 361 works of fiction in French, dated with a reasonably good precision (mean error of 1.71 years per book for books present in the CIDRE corpus). The quality of this corpus can be considered sufficient for it to be used as a reference corpus that serves to account for language change in general. Note that there is a substantial overlap between the content of the CIDRE corpus and our reference corpus (146 novels out of the 361).

4. Testing the rectilinearity hypothesis

In the introduction, we mentioned the rectilinearity hypothesis which posits that some aspects of the idiolect evolve in a linear fashion over time and that this evolution should be detectable. Importantly, the hypothesis does not say that this evolution is relevant for all linguistic features and should affect the same features for each individual. The use of some linguistic features may remain stable, but some evolution should nevertheless be observed for some others, contradicting the conservatist hypothesis (which assumes no linguistic changes). Furthermore, we add the prediction that idiolects evolve constantly and do not return to earlier stages, even if some linguistic features might.

4.1. Methods: Robinsonian Matrices

Robinsonian matrices are distance matrices that have cells whose values increase when moving away from the diagonal. They were introduced in the context of archeological deposits, to study the chronological evolution of the

Table 1. An example of a Robinsonian distance matrix: both ($text_1$, $text_2$) and ($text_2$, $text_3$) are lower than ($text_1$, $text_3$).

	$text_1$	$text_2$	$text_3$
$text_1$	0	2	4
$text_2$		0	1
$text_3$			0

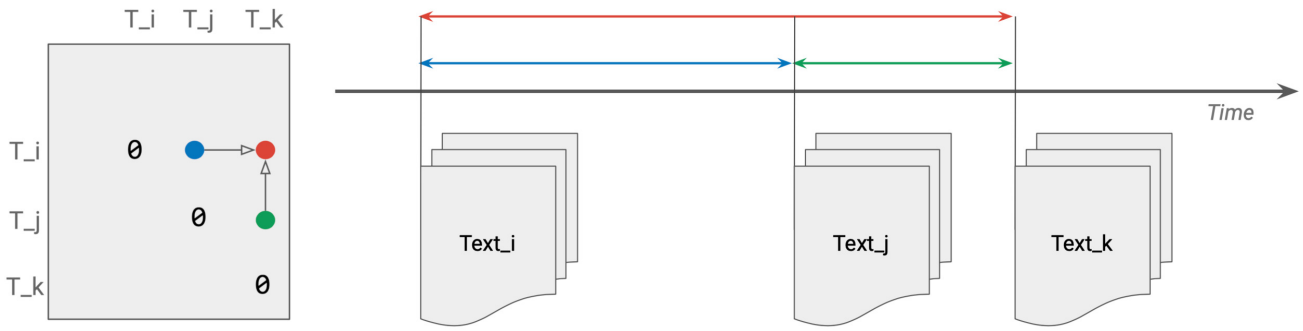


Figure 1. Illustration of the idea of Robinsonian distances between texts.

The colored dots (schema on the left) and arrows (schema on the right) represent distances. If the chronology of the text in the corpus is reflected in the measured distances, we expect that $\max(\delta(text_i, text_j), \delta(text_j, text_k)) \leq \delta(text_i, text_k)$ is true.

style of pottery fragments (W. S. Robinson). More formally, applying this concept to texts, given a matrix δ expressing the distance between novels, we say that δ is Robinsonian if for any set of three distinct texts $text_i$, $text_j$ and $text_k$ such that $\text{date}(text_i) < \text{date}(text_j) < \text{date}(text_k)$,
$\max(\delta(text_i, text_j), \delta(text_j, text_k)) \leq \delta(text_i, text_k)$.

To evaluate the rectilinearity hypothesis on a distance matrix reflecting changes in the idiolect, we measure to what extent the distance matrices corresponding to the texts of the CIDRE corpus are Robinsonian. In order to do this, we can compute the *Robinsonian score*, which we define as the percentage of triples of cells $(\delta(text_i, text_j), \delta(text_j, text_k), \delta(text_i, text_k))$, for which the inequality above is verified.

It is also possible to estimate a *p*-value, i.e. the probability that a Robinsonian score as high as the one being tested could be obtained by chance, by evaluating this score again after randomly changing the order of the texts. Getting a low *p*-value would support the rectilinearity hypothesis.

Before evaluating the rectilinearity hypothesis on the CIDRE corpus, we first used a dated corpus of Maurice Leblanc as a testbed to compare different feature representations for the texts and different ways of measuring distance. We used tokens, characters, lemmas and so-called *motifs* (Legallois et al.) as features. A motif is a sequence of lemmas and POS-tags. As function words tend to be the most relevant features of idiolectal signals (Barlow), grammatical information, i.e. function words and POS-tags, are crucial for the task. However, the tagset of our part-of-speech tagger is not fine-grained enough, losing important information for some categories. For example the difference

Table 2. Examples of unigrams and bigrams and the different types of features

"Il est fâcheux que cela traîne en longueur…"	tokens	characters	lemmas	motifs
unigrams	['Il', 'est', 'fâcheux', 'que', 'cela', 'traîne', 'en', 'longueur', '...']	['I', 'l', ' ', 'e', 's', 't', ' ', 'f', 'â', 'c', 'h', 'e', 'u', 'x', ' ', 'q', 'u', 'e', ' ', 'c', 'e', 'l', 'a', ' ', 't', 'r', 'a', 'î', 'n', 'e', ' ', 'e', 'n', ' ', 'l', 'o', 'n', 'g', 'u', 'e', 'u', 'r', '...']	['il', 'être', 'fâcheux', 'que', 'cela', 'traîner', 'en', 'longueur', '....']	['il', 'être', 'ADJ', 'que', 'cela', 'PRES', 'en', 'NC', '...']
bigrams	[('Il',"est"), ('est',"fâcheux"), ('fâcheux',"que"), ('que',"cela"), ('cela',"traîne"), ('traîne',"en"), ('en',"longueur"), ('longueur',"...")]	['Il', 'l ', ' e', 'es', 'st', 't ', ' f', 'fâ', 'âc', 'ch', 'he', 'eu', 'ux', 'x ', ' q', 'qu', 'ue', 'e ', ' c', 'ce', 'el', 'la', 'a ', ' t', 'tr', 'ra', 'aî', 'în', 'ne', 'e ', ' e', 'en', 'n ', ' l', 'lo', 'on', 'ng', 'gu', 'ue', 'eu', 'ur', 'r...']	[('Il',"être"), ('être',"fâcheux'),('fâcheux',"que"), ('que',"cela"), ('cela',"traîner"), ('traîner',"en"), ('en',"longueur"), ('longueur',"...")]	[('Il',"être"), ('être',"ADJ'),('ADJ',"que"), ('que',"cela"), ('cela',"PRES'), ('PRES',"en"), ('en',"NC'), ('NC',"...")]

between 'un' and 'le' ('a' and 'the') is ignored, and both are tagged as determiners. We therefore used the following strategy: content words were replaced with their POS-tags while function words were replaced with their lemma. This approach allowed us to keep relevant linguistic information, especially at the grammatical level. Legallois et al. proved that these motifs were effective in finding author-specific style characteristics, making it possible to identify interesting examples in corpus studies.

We compared different lengths of n-grams (unigrams to pentagrams) of tokens, characters, lemmas and motifs (see Table 2 for examples of different types of features). The texts of the corpus were represented by the top 500 features with the highest relative frequency. The different distance metrics we used come from stylo R (Eder et al.), in which we entered our corpora. Figure 2 shows the percentage of the distance matrix (calculated for the Leblanc corpus) that is Robinsonian for different feature configurations.

We therefore decided to detect the chronological signal in the corpora of CIDRE and the reference corpus using motif trigrams and the canberra metric as the default option, since trigrams are a medium size and the canberra metric performed slightly better than the others. However, the choice of motifs was motivated by the use of these features in the second series of experiments presented next in this section and by the fact that the lower part of the error bar in Figure 2 is at the highest level of all four tested features. The feature vectors in this experiment contain the scores of the 500 features with the highest relative frequencies.

4.2. Results

The scores for the different authors of the CIDRE corpus and the reference corpus can be found in Table 3. To know whether these scores are meaningful, we compared them with a distribution of random permutations of the distance matrix. For 10 000 random permutations, we calculated the percentage that obtained a Robinsonian score higher than the score of the actual distance

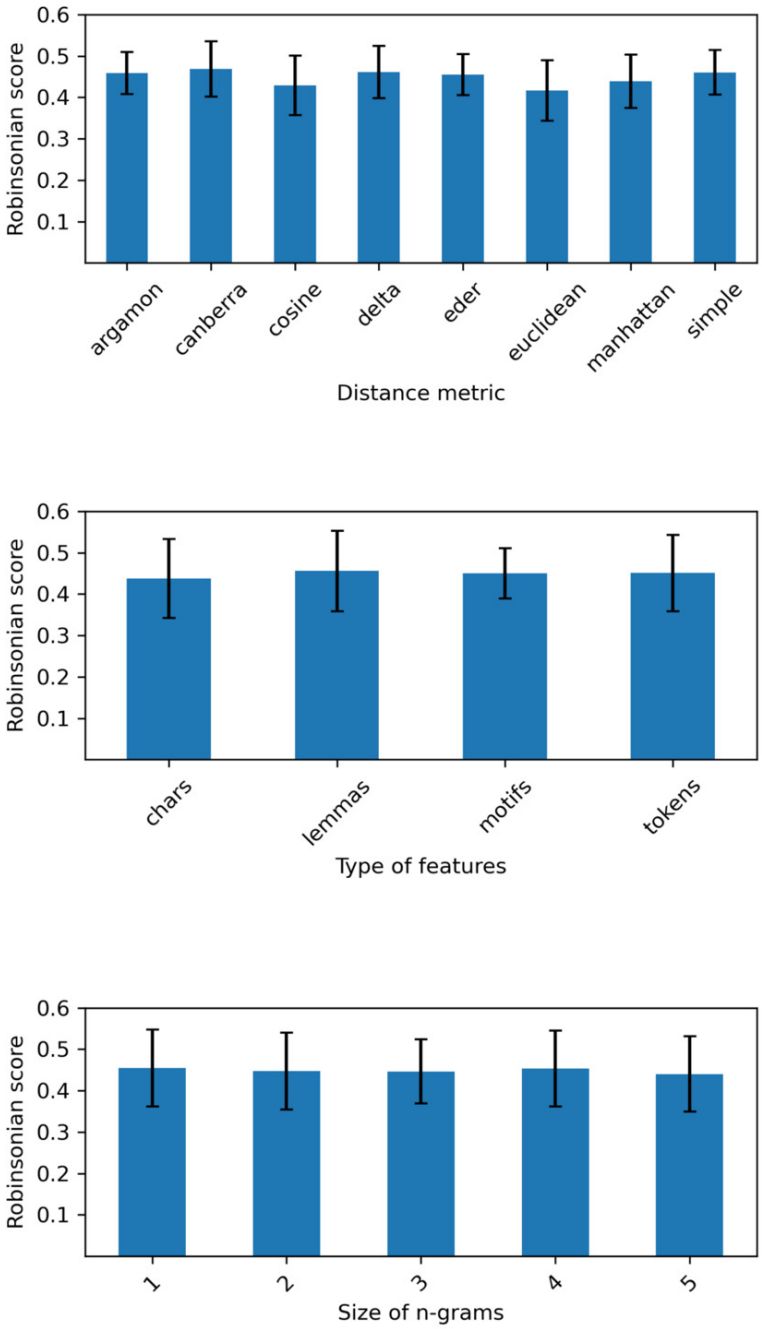


Figure 2. Robinsonian scores for different configurations of features used (eight different distance metrics, four different types of features, five different lengths of n-grams).

Each combination of distance metric, type of features and length of n-gram was tested. Error bars represent the 95% confidence interval of the mean score of all configurations that includes a given parameter. No configuration is significantly better than others. The highest result of 0.50 is obtained using quadrigrams of lemmas with the canberra metric, but this experiment does not allow us to identify a combination of features that is significantly the best to capture the chronological signal in the data.

matrix. Table 3 demonstrates that the distance matrices obtained are significantly more Robinsonian for all the authors than random permutations, except for Comtesse de Ségur.

Table 3. The Robinsonian scores for the authors in CIDRE and the reference corpus, followed by the probability of obtaining these scores by chance if the chronological signal in the data is absent.

Corpus	Robinsonian score	Probability of Robinsonian score if no chronological signal is present in the data
Comtesse de Ségur	0.38	0.14
Daniel Lesueur	0.41	0.00
Pierre-Alexis Ponson du Terrail	0.41	0.00
Gustave Aimard	0.42	0.01
Honoré de Balzac	0.44	0
Michel Zévaco	0.46	0
Jules Verne	0.47	0
George Sand	0.49	0
Paul Féval	0.49	0.00
Henry Gréville	0.62	0
Émile Zola	0.63	0
Reference Corpus	0.34	0

Decimal numbers come from rounding and plain zeros have not been rounded.

4.3. Discussion

First, it should be noted that the percentage of Robinsonian cells in the matrix is dependent on the number of works in the corpus. Larger corpora lower the probability of getting Robinsonianness by chance. Therefore, the score of 0.34 for the reference corpus seems low, but is actually very high for this number of works. Second, it should be kept in mind that only the absolute order of works plays a role in our method and that it does not take into account the exact difference in years between works. That is to say: if $\max(\delta(text_i, text_j), \delta(text_j, text_k)) \leq \delta(text_i, text_k)$ is false, it does not matter how much more $\max(\delta(text_i, text_j), \delta(text_j, text_k))$ is than $\delta(text_i, text_k)$.

The fact that different types of features produce similar results on the Maurice Leblanc corpus is not that unexpected regarding the literature. Stamou identified a number of stylistic markers that were of interest in many stylochronometric studies, namely: punctuation, characters, part of speech tags, most common words including function words, frequencies of selected content words, hapax and vocabulary richness. She suggested that there might not be a "single universal stylochronometer" that can apply to every corpus.

The results from our experiments show that there is a strong chronological signal in the data, except for the corpus of Comtesse de Ségur. A possible explanation for this exception could be that this corpus is too small, representing only 3.8% of the total tokens in CIDRE. Another explanation is that this corpus might be heterogeneous, as it includes children's stories, bible stories for children and fairy tales. However, in general our results are in line with the rectilinearity hypothesis: the style of an author generally evolves smoothly over time. No regression (texts stylistically similar to earlier texts) can be observed.

In the next section, we discuss our second series of experiments in which we trained a linear regression model to automatically predict the date of writing of various novels from our reference corpus.

5. Predicting Year of Writing using Linear Regression

In this section, we first examine the chronological evolution of the idiolect (and the reference corpus) by training models on the corpora of idiolects in CIDRE and then predicting the year of writing of different works using cross-validation. The hypothesis is simple: if this type of experiment is successful, the results are in favor of the rectilinearity hypothesis. In other words, the frequency of some linguistic forms increases or decreases in a linear fashion to such an extent that we can detect the year of writing. Second, we do not just want to verify if there is a chronological signal, but also if we can identify the linguistic material at the heart of this evolution. Therefore, we will present a feature selection method that identifies the features that change the most in frequency over time. These features will be used for the qualitative study in section 6. Furthermore, besides these hypotheses and goals, we also have the general objective of proposing new ways to evaluate stylometric methods. For this purpose, we will have recourse to the state of the art literature on linear regression models and verify that it can be used in the stylochronometry context.

5.1. Methods: Regression Models

Various previous studies have used regression techniques in order to date literary works. For example, Frischer used regression techniques (among other methods) to date the *Ars Poetica* of Horace. However, by today's standards, the number of features in this regression was very low so we will mainly discuss more recent work. A representative study using regression is Klaussner and Vogel work from 2018 (henceforth K&V). They used it in a machine learning task that consisted in predicting the year of writing of a work, focusing on two corpora in English: the work of Henry James and that of Mark Twain. They also used the years 1860-1919 of The Corpus of Historical American English (COHA; Davies et al.) as a reference corpus to capture the 'general' language in North America at the time, to check whether the changes detected in the work of James and Twain were shared by the community or were idiosyncratic. Four types of features were considered: character n-grams, part-of-speech tags, word stems, and lemmas with POS-tags; for each of them unigrams to quadrigrams were tested. This resulted in a total of 32 models (2 authors, 4 types of features and 4 types of n-grams). On each model, the elastic nets algorithm was applied to reduce the number of parameters. The models were evaluated using the measure of root mean squared error (RMSE), which reflects the difference (measured in years) between the prediction and the real year of writing. At this point, note that one should keep in mind that this metric is quite sensitive to outliers, as the error is squared. A baseline performance was also measured "*by using the mean of the data for the prediction of every instance*"; meaning that every work that was dated by the baseline received the same prediction (the

mean of all training instances). This would correspond to a model that has a R^2 score of 0 (Field). The best results were obtained by K&V using lemmas with POS-tags in unigrams and bigrams.

Although our experiments share many similarities with K&V, we made some different choices for our models. First, we used only motifs consisting of n-grams (unigram to pentagram, but all incorporated in the same model instead of different models as in K&V). The notion of motif is anchored in previous studies: it has been shown that they are helpful through qualitative analysis (Legallois et al.). Second, the feature section algorithm we chose is Lasso LARS (Efron et al.) with cross validation of 5 (80% training, 20% testing) and not elastic nets. We chose this algorithm because our aim was not to find the most compact model, unlike K&V, but a model that drastically reduces the number of features so that they can be inspected manually (see our qualitative study, Section 6 of this paper). Moreover, as a selection criterion of features, we require that the features be present in at least 20% of an author's texts, whereas in the work of K&V, features had to be present in all data points. This much lower threshold was chosen here because we think it is possible — at least theoretically — for a language innovation to be totally new or for some structures to entirely disappear. Also, K&V concatenated texts written in the same year into one data point by putting the texts behind each other in the same file, whereas we kept them as separate data points with the same value for the year, since we believe that this better represents the data. However, to ensure comparability with K&V, we decided to measure the RMSE and the RMSE-baseline for our experiments. We also compared our results to our reference corpus, and tried the algorithm of elastic nets,[1] as well as elastic nets cross validated.[2] In the end, however, we found that Lasso LARS cross validated performed much better on most of our corpora and that the number of features it selected was better suited for qualitative studies (the elastic nets selected either no features or thousands of features, making a qualitative study impossible). Details about the comparison of feature selection algorithms can be found in the supplementary material.[3]

5.2. Results

For every author, we measured the correlation between the actual year and the predicted year and the value of R^2 (expressed between 0 and 1), which represents the amount of variation of the data that is explained by the model (Field). The results can be found in Table 4 and Figure 3. Excellent results were obtained for Jules Verne, Émile Zola, George Sand, Henry Gréville, Daniel-Lesueur and Honoré de Balzac: the models (selected n-grams of motifs) were capable of predicting the large majority of the variation in the data. The models

1 https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html

2 https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNetCV.html#sklearn.linear_model.ElasticNetCV

3 The filename is: 'results_LassoLars_vs_ElasticNet.txt'.

Table 4. The regression experiment was very successful in explaining the variance of the corpora in gray, and considerable for the other corpora, except for Pierre Alexis Ponson du Terrail, where it was inefficient.

Author	Correlation	R^2	#β	RMSE	RMSE-b	Remarks
Jules Verne	0.94	0.89	57	3.91	11.83	
Daniel-Lesueur	0.92	0.84	14	3.39	8.46	
Émile Zola	0.92	0.83	34	4.50	11.02	
Honoré de Balzac	0.90	0.78	42	2.44	5.26	
George Sand	0.88	0.77	61	6.13	12.78	
Henry Gréville	0.78	0.55	31	2.85	4.27	Regressors in active set degenerate (1/5 folds)
Michel Zévaco	0.75	0.55	23	3.52	5.22	ConvergenceWarning: Regressors in active set degenerate (2/5 folds)
Gustave Aimard	0.70	0.49	21	5.96	8.21	
Paul Féval	0.51	0.26	17	8.72	10.14	ConvergenceWarning: Regressors in active set degenerate (3/5 folds)
Comtesse de Ségur	0.45	0.18	18	3.57	3.96	ConvergenceWarning: Regressors in active set degenerate (1/5 folds)
Pierre Alexis Ponson du Terrail	-0.04	-0.55	10	5.69	4.57	
Reference corpus	0.84	0.70	208	11.30	20.50	

explained a substantial amount of variation in the data for the authors Michel Zévaco, Gustave Aimard, la Comtesse de Ségur and Paul Féval, but less than half of it. Lastly, for Pierre Alexis Ponson du Terrail, the model was not able to explain any variance in the data, and thus the experiment was not successful at all. The same observations can also be made by comparing the evaluation metric root mean squared error (RMSE) and the baseline metric (RMSE-baseline) put forward by K&V.

It is important to mention that the modelling does not always (completely) converge for a given K-fold. This problem is mostly noticeable for Paul Féval. There seems to be a relation between the performance and this issue, but convergence cannot explain the poor performance of the model on the corpus of Pierre Alexis Ponson du Terrail: all the models on this corpus converged. We also had a look at which works were well predicted and which ones were outliers. See for example Figure 4, where it can be seen that *L'Auberge des Saules* by Daniel-Lesueur was predicted about 6 years too late, but *Comédienne* exactly at the time it was written. Other figures in the same style can be found in the supplementary material (directory *plots_regression_par_auteur*).

5.3. Discussion

The average proportion of each author in our whole corpus is 9% since we have 11 authors. Our results show that larger corpora (Sand: 15.3% of the whole corpus, Verne: 14%, Zola: 13% and Zévaco: 10.5%) perform very well and that smaller corpora obtain lower scores (Ségur: 3.8%, Aimard: 5.4%, Féval: 6.4%). This suggests that Petré and Van de Velde are right when they say that corpus size matters a lot, even if it does not explain why our worst performing corpus

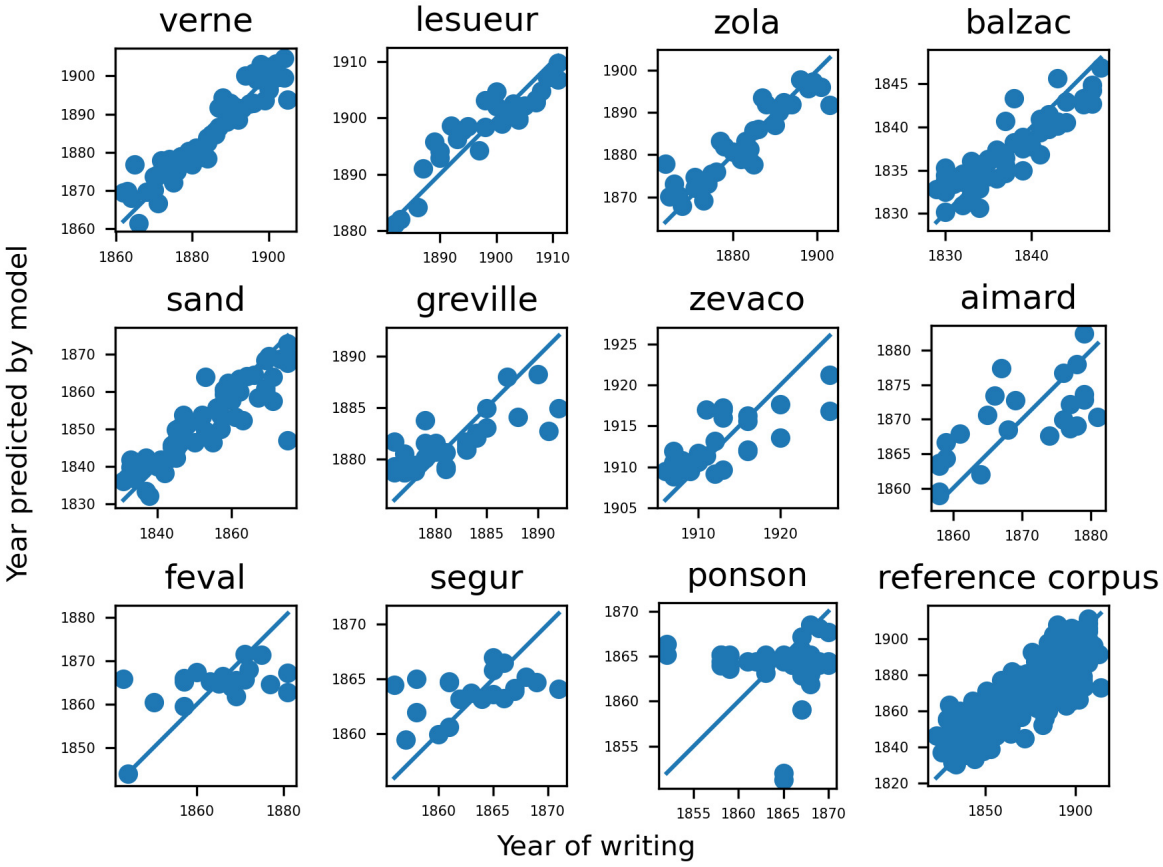


Figure 3. The results of the regression experiment for all corpora in CIDRE, sorted by performance (left to right, top to bottom).

The blue line represents a perfect correlation.

is Pierre Alexis Ponson du Terrail, which is a medium size corpus (representing 9.1% of the total size of CIDRE). However, it is possible of course that we are not aware of certain circumstances in the publication process of the different authors that might explain these results or that the dating of this corpus is of lesser quality. For example, the literary work of Ponson du Terrail is less well known than that of some other writers in CIDRE: his works were dated using information mostly from Wikipedia, which is not as reliable a source as those used for other authors. If the poorer dating of some novels by this writer is the source of the failure of our model, it means that our method is sensitive to individual data points. Indeed, going back to the previous experiment and Table 3, we see that there is a highly significant chronological signal for this corpus, which means that the approach works globally and that specific cases of failure should be further investigated.

However that may be, for most of the authors we get a very high value of R^2, which means that the chronology can explain almost all the variance of the models. This is confirmed by the fact that the value of RMSE is much lower than the RMSE-baseline value. We can thus conclude that the results on all our corpora minus one are in line with the rectilinearity hypothesis.

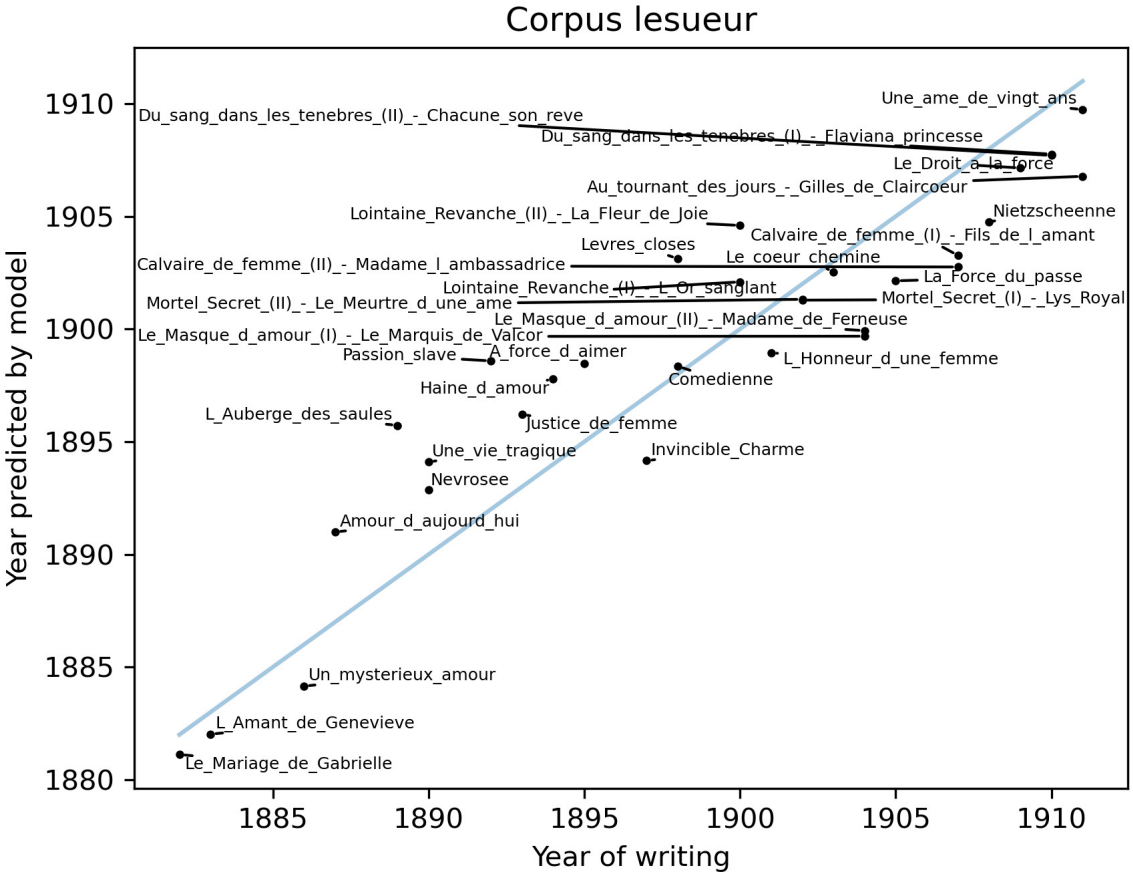


Figure 4. The result of the regression experiment for the corpus of Daniel-Lesueur with annotated data points on the scatterplot.

The blue line represents a perfect correlation.

A second interesting result that we obtained is the number of features selected per model (column #β in Table 4). For all the corpora in CIDRE the number lies between 10 and 61, which are numbers that make it possible to examine all the features of a corpus in a qualitative study. A direct comparison with K&V is difficult, as we did not work on the same corpora. However, we observe that the number of predictors per model (column #β in Table 4) has a smaller range for the different models we developed (the models of K&V range from counting 1 predictor to 315). Nevertheless, it should be kept in mind that our feature selection algorithm probably removes correlated features and that the random seed plays a role in which ones. Therefore, for a complete qualitative study of one author, it might be worthwhile repeating the regression experiment a number of times with different random seeds.

The features that characterize the evolution of an author do not necessarily have to be frequent. While some features are quite frequent, such as the increasing motif 'y' (anaphor referring to a prepositional phrase beginning with the preposition 'à') found for Zola, which obtains the relative frequency of 0.0027 at its maximum, some others, such as the decreasing motif 'autrui' (other people), have a low relative frequency, 0.0008 at its maximum. This

conclusion is similar to a finding in Koppel and Schler, who found that idiosyncratic features that play an important role in authorship attribution also tend to be of low frequency. How we explore the features of the models will be discussed in the next section.

6. A Qualitative Study of some Motifs Sensitive to Diachronic Change

In this section, we look more closely at features selected by the regression algorithm (presented in the previous section). These features, or motifs, are at the heart of idiolectal evolution and are assumed to be easily interpretable. Let's examine if this is true.

6.1. Methods: Manual Inspection

We scrutinized the motifs relevant for four authors on whom our models obtained good results: Balzac, Daniel-Lesueur, Sand and Zola (looking deeper into the motifs attached to authors for whom we obtained poor results would not make much sense). For these authors, we inspected whether the selected motifs were interpretable by looking at examples from the corpus in context, to see if they corresponded to meaningful linguistic patterns.

First, it is clear that some forms are not interpretable: in Balzac, for example, there is a decrease in the use of adverbs over time, but the adverbial category is relatively heterogeneous so that it is difficult to interpret this phenomenon. The same can be observed with the motif "NC_,_avoir" (also in Balzac) which increases; this motif is realized in sequences (*patronne, avait* - mistress, had; *frère, ont* -brother, had; *ciel, as* - heaven, have) of which, at first sight, nothing really relevant can be said. Another group of difficult motifs in Balzac is the more frequent use of the pronoun "*on*" ("on PRES", ". on", "ADJ que on"). These patterns are hard to interpret, especially since the pronoun "*on*" has a fairly wide range of referential values.

Another example from Zola concerns verbs such as *bouleverser* (to upset) and *convaincre* (to convince), which have an increasing frequency over time. But here again, there is no immediate explanation for this usage. However, we were pleased to see that most motifs are interpretable (for example, we estimated that about three quarters of the motifs retained for Zola were). In the rest of this section, we will discuss some of the (groups of) motifs that we found interesting, or that have been previously noticed by researchers of the field of stylistic analysis.

6.2. Results

A number of interpretable motifs can be considered as stylemes. For example, in Zola, there is a set of motifs organized around ". Et" (the conjunction "and" at the beginning of the sentence): ". Et le NC être" ; ". Et, dès" ; ". Et ce être" whose use increases, as shown in example 1:

> (1) Quoi donc ? Était-ce la fin ? Un souffle glacé avait couru sur le camp, anéanti de sommeil et d'angoisse. **Et ce fut** alors que Jean et Maurice reconnurent le colonel de Vineuil [...]
> (Zola, *La débâcle*)
>
> What then? Was it the end? An icy breath had run over the camp, annihilated by sleep and anguish. **And it was** then that Jean and Maurice recognized Colonel de Vineuil [...]

This use (called the revival "et") was noticed very early by stylisticians (Thibaudet; but see Bordas and Badiou-Monferrand for modern accounts). It is not considered an idiosyncrasy, since this motif was also used by Flaubert, who can be said to have been imitated by Zola (Thibaudet; Gauthier). Flaubert considered that it was "an old biblical tic which is annoying".

Another set of motifs is, meanwhile, on the decline, including "NCCOR", a tag for parts of the human body, which are used in the physical description of the characters (*épaules; tête; main; yeux*, etc. - shoulders; head; hand; eyes, etc.). It is difficult to interpret the reason for this decrease; perhaps the form, or at least its repetition, was considered a cliché by the author.

In the same way, in Sand, we also notice that motifs linked to units referring to parts of the body tend to decrease, for example "NCCOR avec NCABS": *et se jeta dans mes **bras avec joie**; Suzanne baissa la **tête avec embarras**...* and threw herself into my **arms with joy**; Suzanne lowered her **head in embarrassment**. This motif associates a movement of the body with a feeling. Again, this change could be considered to be due to the avoidance of a cliché, but this is a hypothesis that will have to be verified in further analysis.

Among the positive motifs of Sand, we note these two forms (", et, comme", ", et, si ce") which share the same rhythmic pattern (see examples 2 and 3). Without going into detail, these patterns may highlight the subordinate sentence by a kind of tension (↗), while the main phase is constructed in detension (↙).

> (2) Après avoir fait quelques tours sous les galeries, il se crut assez calme pour retourner à l'atelier, **et, comme** il redescendait l'escalier des Géants, il se trouva tout à coup face à face avec le Bozza.
> (Sand, *Les Maîtres mosaïstes*)
>
> After having taken a few turns under the galleries, he believed himself calm enough to return to the workshop, **and, as** he went back down the staircase of the Giants, he found himself suddenly face to face with the Bozza.

> (3) Dès lors, j'espérais qu'elle pourrait aimer Narcisse, **et, si cet** excellent jeune homme pouvait être heureux par elle, c'était à la condition de ne plus souffrir du passé. (Sand, *Narcisse*)
>
> From then on, I hoped that she would be able to love Narcisse, **and, if this** excellent young man could be happy thanks to her, it was on the condition of not suffering from the past anymore.

For Balzac, we found the decreasing motif "tout_à_NC" which corresponds in the vast majority of cases to the adverb "tout à coup" *all of a sudden*. Again, we suspect that the decrease of this motif could be caused by the avoidance of clichés. An interesting example of increasing motifs of Balzac is the motif "dire_à_NP" *"say to Proper Name"*. When inspecting the corpus, we noticed that this phrase is used in different ways: sometimes it is inserted inside a dialogue as illustrated in example (4), often it is used to mark the transition from narration to dialogue as in (5) and vice-versa, as in (6). We consider it a stylistic means to dynamize the switches between narratives and dialogues. Often this construction is used to put a long grammatical subject after the verb and direct object (as in 4 and 6), which also creates a stylistic effect.

> (4) Il ne faut pas demander à monsieur pourquoi il vient, **dit à Castanier** une vieille portière, vous ressemblez trop à ce pauvre cher défunt.
> (Balzac, *Melmoth reconcilié*)
>
> One shouldn't ask this gentleman why he came, **said** an old doorkeeper **to Castanier**, you look too much like the poor, dear deceased.
>
> (5) Le commandant, qui l' étudiait, s'apercevant de cette insensibilité, **dit à Gérard** : Le serin n'en sait pas long.
> (Balzac, *Les Chouans*)
>
> The commander, who was studying him, and noticed this insensitivity, **said to Gérard**: The fool does not know much.
>
> (6) J'attends la réponse, **dit à Rastignac** le commissionnaire de madame de Nucingen.
> (Balzac, *Le père Goriot*)
>
> I'm waiting for an answer, **said** the commissioner of Madame de Nucingen **to Rastignac**.

Finally, for Daniel-Lesueur, it is worth mentioning the increasing motif "..._DETPOSS_NC_..." (see examples 7 and 8), by which a noun preceded by a possessive determiner in between two ellipsis punctuation marks dramatizes reported thoughts and speech by invoking a close relation *mon enfant; ma soeur; mon amie* (*my child; my sister; my friend*).

> (7) Ah ! ma mère **... ma mère ...** pensait Hervé, [...]
> (Daniel-Lesueur, *Le Masque d'Amour II - Madame de Ferneuse*)
>
> Ah ! my mother **... my mother...** thought Hervé, [...]
>
> (8) Je suis perdue ! ... Perdue ! **... Ma chérie ...** Invente quelque chose ! ... Ah ! sauve-moi !
> (Daniel-Lesueur, *Justice de femme*)
>
> I'm lost! ... Lost! **... My darling...** Think of something! ... Ah! save me!

6.3. Discussion

As already mentioned, not all the motifs identified automatically are interpretable. Many, however, are stylistic in nature without it being possible to determine whether these uses are a deliberate choice by the author, or whether they are a form of automatism. To shed light on this question, a more precise analysis involving literary expertise should be undertaken. Our analysis provides the literary scholar, the stylistician and the linguist with statistically relevant evidence of the evolution of certain forms. It is up to these specialists to show correlations between forms, to propose interpretations. This type of approach can provide an empirical basis for more theoretical research (Philippe). Our hope is to have demonstrated that our method, which combines the use of motifs and the feature selection method of Lasso LARS, identifies a large number of stylistically interesting patterns and can be a useful tool in the qualitative analysis of the evolution of the idiolect.

7. General Discussion and Future Work

7.1. Contribution of the Work

In this article we investigated the chronological evolution of the idiolect. We examined whether support could be found for the rectilinearity hypothesis which states that the evolution of the idiolect is rectilinear in time, and whether the linguistic structures at the heart of idiolectal change could be identified. Using the Corpus for Idiolectal Research (CIDRE), we developed two methods that could help reach these goals. First, we introduced the idea of evaluating to what extent the distance matrices of works of one author are robinsonian. For ten out of eleven corpora in CIDRE, we found that the Robinsonian score was significantly high, suggesting that chronology plays a crucial role in the idiolect of an author. Second, we developed linear regression methods to predict the year of writing of a work and selected linguistic features that are key in the process of idiolectal change. We found that the majority of regression models were highly successful, again supporting the rectilinearity hypothesis. Third, these models allowed us to find a number of features (in the form of *motifs*) that lent themselves to manual examination in a qualitative study, demonstrating both the usefulness of these features and the validity of our methods. We believe that the use of motifs is complementary to the use

of lemmas and tokens. As for example Brunet illustrates in his study of the vocabulary used by Zola, using lemmas allows a researcher to interpret the topics of a writer. In the present study we demonstrate that motifs, on the other hand, might give more insights in stylistic forms.

We believe that working on the concepts of idiolect and chronological change can have an impact on related research themes. Modeling the idiolect could be useful, for example for the task of automatic text dating that was included in the 2015 SemEval campaign: *Task 7: Diachronic Text Evaluation* (Popescu and Strapparava). A corpus of snippets from newspaper articles dating from 1700 until 2010 was composed and the task consisted in dating these snippets. It could be interesting to see if the idiolect plays a role and if it can enhance the classification results.

A theme for which the concept of chronological variation could be interesting is authorship attribution and authorship verification, which involves checking whether a pair of documents are written by the same person (Kestemont et al.). Nowadays, the chronology of the writing is not taken into account; only the idiolect of each author in the corpus is modeled. It is quite possible, however, that taking the date of writing into consideration would enhance the modeling. Many different features have been explored to model the idiolect of authors for this task: n-grams of words or characters (e.g. Stamatatos; Antonia et al.; Sari et al.), syntactic structures (Sundararajan and Woodard; Zhang et al.) and even discourse structure (Ding et al.) but we are not aware of models that take idiolectal variance over time into account. However, especially for writers with long careers, it could be meaningful.

In this study, we focused on methods and on the evaluation of results. We argue that the use of standard corpora, baselines and evaluation metrics could help enhance the comparability of studies in the field of stylometry and that this would help the research community gain greater insight into the robustness of the results. In our experiment on the Robinsonian matrices, we used random results as a baseline. For research questions that have not yet been addressed in the literature, this is a useful starting point, as shown in the work of Bulteau et al., who developed two algorithms to estimate the probability that a tree produced by a hierarchical clustering algorithm — for instance produced by stylo R (Eder et al.) — reflects a chronological order by chance. In our experiment using regression models, we compared our methods with those of Klaussner and Vogel from their 2018 publication, using their baseline RMSE and the standard baseline of regression models, R2 (Field).

An important contribution of this study is that it addresses questions of evaluation. We have seen that the development of off-the-shelf-packages has made it possible to shed new light on long-standing research questions. For example Schmidt-Petri et al. used the rolling-classification algorithm from stylo R (Eder et al.) to examine the contribution of Harriet Taylor Mill to the essay *On Liberty*, which is officially contributed solely to John Stuart Mill,

her husband. They found that there is stylometric evidence that she should indeed be considered a co-author of the work. However, as stylo R does not enable any statistical evaluation of the classification results, the authors had no straightforward means of interpreting their reliability and had to undertake considerable extra work to estimate the robustness of the results. We therefore think that working on the question of evaluation of stylometric methods is a topic in the field of stylometry that needs to be developed further and we hope to have made a useful contribution to it.

7.2. Future Work

The most obvious future work that should result from this study is a detailed qualitative analysis of the selected motifs in CIDRE, for which the regression models obtain good results. These studies should also contain a detailed comparison with the reference corpus in order to decide if the observed change can be interpreted as a rather general diachronic language change or an idiosyncrasy, using for example the methods that Mollin used to identify idiosyncratic collocations. This should be done, however, in collaboration with literary experts of the writers in question in order to compare the findings of the method with what is already known in the field of stylometry and stylistics. In addition to the identification of idiosyncratic motifs, collaborations with literary experts would allow us to get a more precise interpretation of the motifs of an author. Indeed, we could for example examine the role that dialogs, narratives, and descriptions play when experts provide us with theoretically and empirically motivated hypotheses on specific authors.

Another straightforward direction for future work is to repeat our experiments on other text genres, for example drama or correspondence. We are considering trying our methods on plays, for example by using the *Théâtre Classique* corpus of Fièvre. However, as theatrical works might be influenced/written by the actors in the plays, the idiolectal signal of the playwright may not be as strong as for works of fiction. Correspondence could be interesting in order to investigate idiolectal changes with respect to the addressee of the letter. We could for example use the Corpus of Early English Correspondence (Raumolin-Brunberg and Nevalainen) and the correspondence of George Sand. Another advantage of using correspondence is that dating letters might be more precise than dating works of fiction. However, it would probably result in corpora that are significantly smaller than the corpora of the authors in CIDRE. As we suspect a strong relation between corpus size and the statistical power of the experiments, the success is not guaranteed for smaller corpora. But on the other hand, the number of letters per author is probably higher than the number of books in CIDRE, which could enhance the statistical power.

A third direction for future work is to evaluate how different people influence each other with their idiolects. Evans investigated how the idiolect of Queen Elizabeth I was influenced by others. It could be interesting to develop a methodology on how influence could be established between authors or even between literary movements.

8. Conclusion

Our experiments demonstrate that there is a significant evolution of the idiolect during an author's lifetime. Our experiments also suggest that some features evolve in a rectilinear manner, steadily increasing or decreasing as the years go by. These features are sufficiently clear-cut to be used to date the year of writing of a book very accurately. We therefore conclude that we found strong support for the rectilinearity hypothesis and that the evolution of the idiolect is a relevant type of intrapersonal variation that exists alongside the strong signal of interpersonal variation. We thus dismiss the proposal that idiolects are stable over time, even though it is true that not all linguistic features evolve. A second contribution of our article is the development of new methods for which we have demonstrated the usefulness in 1) assessing the chronological signal of the idiolect in corpora and 2) identifying linguistic structures that are at the heart of this evolution. These features can in turn be used for qualitative studies with stylistic objectives.

Peer-Reviewers: Simon DeDeo, David Mimno

All scripts, supplementary materials and data used for our experiments are available in the online Harvard Dataverse directory: https://doi.org/10.7910/DVN/WCMZOK, except for the CIDRE corpus, that is freely available in the following Zenodo repository: https://doi.org/10.5281/zenodo.4707812.

Acknowledgments

We thank the two reviewers for their insightful comments and suggestions. This work has been developed in the framework of the IRN (International Research Network) Cyclades (Corpora and Computational Linguistics for Digital Humanities). This work was also supported in part by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute) and reference ANR-16-IDEX-0003 (I-Site Future, programme "Cité des dames, créatrices dans la cité").

Submitted: February 02, 2022 EDT, Accepted: April 26, 2022 EDT

REFERENCES

Anthonissen, Lynn, and Peter Petré. "Grammaticalization and the Linguistic Individual: New Avenues in Lifespan Research." *Linguistics Vanguard*, vol. 5, no. s2, June 2019, doi:10.1515/lingvan-2018-0037.

Antonia, A., et al. "Language Chunking, Data Sparseness, and the Value of a Long Marker List: Explorations with Word n-Grams and Authorial Attribution." *Literary and Linguistic Computing*, vol. 29, no. 2, May 2013, pp. 147–63, doi:10.1093/llc/fqt028.

Badiou-Monferrand, Claire. "Rémanence Des Et de Relance En Français Moderne et Contemporain: Du 'Résidu' Au 'Reliquat.'" *Le Français Moderne*, vol. 88, no. 2, 2020, pp. 295–312.

Barlow, Michael. "Individual Usage: A Corpus-Based Study of Idiolects." *Proceedings of LAUD Conference*, 2010.

Bloch, Bernard. "A Set of Postulates for Phonemic Analysis." *Language*, vol. 24, no. 1, Jan. 1948, pp. 3–46, doi:10.2307/410284.

Bordas, Éric. "Et La Conjonction Resta Tensive. Sur Le et de Relance Rythmique." *Français Moderne*, vol. 73, no. 1, 2005, pp. 23–39.

Brainerd, Barron. "The Chronology of Shakespeare's Plays: A Statistical Study." *Computers and the Humanities*, vol. 14, no. 4, Dec. 1980, pp. 221–30. *Crossref*, doi:10.1007/bf02404431.

Brooke, Julian, et al. "GutenTag: An NLP-Driven Tool for Digital Humanities Research in the Project Gutenberg Corpus." *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, 2015, pp. 42–47, doi:10.3115/v1/w15-0705.

Brunet, Etienne. *Le Vocabulaire de Zola*. Slatkine, Champion, 1985.

Bulteau, Laurent, et al. "Reordering a Tree According to an Order on Its Leaves." *33rd Annual Symposium on Combinatorial Pattern Matching (CPM 2022)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022. *Google Scholar*, doi:10.4230/LIPIcs.CPM.2022.24.

Can, Fazli, and Jon M. Patton. "Change of Writing Style with Time." *Computers and the Humanities*, vol. 38, no. 1, Feb. 2004, pp. 61–82, doi:10.1023/b:chum.0000009225.28847.77.

Church, Kenneth, et al. "Using Statistics in Lexical Analysis." *Lexical Acquisition: Exploiting on-Line Resources to Build a Lexicon*, Psychology Press, 1991, pp. 115–64.

Cox, D. R., and Leonard Brandwood. "On a Discriminatory Problem Connected with the Works of Plato." *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 21, no. 1, Jan. 1959, pp. 195–200. *Crossref*, doi:10.1111/j.2517-6161.1959.tb00329.x.

Craig, Hugh. "Stylistic Analysis and Authorship Studies." *A Companion to Digital Humanities*, vol. 3, 2004, pp. 233–334.

Cropp, Martin, and Gordon Fick. "Resolutions and Chronology in Euripides: The Fragmentary Tragedies." *Bulletin Supplement (University of London. Institute of Classical Studies)*, 1985, pp. iii–92.

Daelemans, Walter. "Explanation in Computational Stylometry." *Computational Linguistics and Intelligent Text Processing*, edited by Alexander Gelbukh, vol. 7817, Springer Berlin Heidelberg, 2013, pp. 451–62. *Crossref*, doi:10.1007/978-3-642-37256-8_37.

Davies, Mark, et al. "The 400 Million Word Corpus of Historical American English (1810–2009)." *English Historical Linguistics 2010: Selected Papers from the Sixteenth International Conference on English Historical Linguistics (ICEHL 16), Pécs, 23-27 August 2010*, vol. 325, John Benjamins Publishing, 2012, pp. 231–62, doi:10.1075/cilt.325.11dav.

Derks, Peter L. "Clockwork Shakespeare: The Bard Meets the Regressive Imagery Dictionary." *Empirical Studies of the Arts*, vol. 12, no. 2, July 1994, pp. 131–39. *Crossref*, doi:10.2190/h489-jh64-lq8c-l4t1.

Devine, A. M., and Laurence D. Stephens. "A New Aspect of the Evolution of the Trimeter in Euripides." *Transactions of the American Philological Association (1974-)*, vol. 111, 1981, p. 43. *Crossref*, doi:10.2307/284118.

Ding, Steven H. H., et al. "Learning Stylometric Representations for Authorship Analysis." *IEEE Transactions on Cybernetics*, vol. 49, no. 1, Jan. 2019, pp. 107–21. *Crossref*, doi:10.1109/tcyb.2017.2766189.

Dittmar, Norbert. "Explorations in 'Idiolects.'" *Amsterdam Studies in the Theory and History of Linguistic Science Series 4*, 1996, pp. 109–28.

Dunning, Ted E. "Accurate Methods for the Statistics of Surprise and Coincidence." *Computational Linguistics*, vol. 19, no. 1, 1993, pp. 61–74.

Eder, Maciej, et al. "Stylometry with R: A Package for Computational Text Analysis." *The R Journal*, vol. 8, no. 1, 2016, doi:10.32614/rj-2016-007.

Efron, Bradley, et al. "Least Angle Regression." *The Annals of Statistics*, vol. 32, no. 2, Apr. 2004, pp. 407–99, doi:10.1214/009053604000000067.

Evans, Mel. *Aspects of the Idiolect of Queen Elizabeth I: A Diachronic Study on Sociolinguistic Principles*. University of Sheffield, 2011.

Field, Andy. *Discovering Statistics Using SPSS: Book plus Code for E Version of Text*. SAGE Publications Limited, 2009.

Fièvre, Paul. "Théâtre Classique." *Université Paris-IV Sorbonne Http://Www. Theatreclassique. Fr*, 2007.

Fleury, Serge, and Maria Zimina. "Trameur: A Framework for Annotated Text Corpora Exploration." *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, 2014, pp. 57–61.

Forsyth, R. "Stylochronometry with Substrings, or: A Poet Young and Old." *Literary and Linguistic Computing*, vol. 14, no. 4, Dec. 1999, pp. 467–78. *Crossref*, doi:10.1093/llc/14.4.467.

Frischer, Bernard. *Shifting Paradigms New Approaches to Horace's Ars Poetica*. 1991.

Gauthier, E. Paul. "Zola as Imitator of Flaubert's Style." *Modern Language Notes*, vol. 75, no. 5, May 1960, p. 423, doi:10.2307/3039860.

Guaresi, Magali, et al. "Entre Rupture et Continuité, Le Discours Du PCF (1920-2020)." *Histoire & Mesure*, vol. XXXVII–1, no. 2, Dec. 2021, pp. 125–62, doi:10.4000/histoiremesure.14904.

Heck, Richard. "Idiolects." *Content and Modality: Themes from the Philosophy of Robert Stalnaker*, Oxford University Press on Demand, 2006, pp. 61–92.

Heiden, S., et al. *Manuel de TXM, Version 0.7.9*. ENS de Lyon & Université de Franche-Comté, 2018, http://textometrie.ens-lyon.fr/files/documentation/Manuel%20de%20TXM%200.7%20FR.pdf.

Jackson, MacD. P. "Pause Patterns in Shakespeare's Verse: Canon and Chronology." *Literary and Linguistic Computing*, vol. 17, no. 1, Apr. 2002, pp. 37–46. *Crossref*, doi:10.1093/llc/17.1.37.

Jaynes, Joseph T. "A Search for Trends in the Poetic Style of WB Yeats." *ALLC Journal*, vol. 1, 1980, pp. 11–18.

Kestemont, Mike, et al. "Overview of the Cross-Domain Authorship Attribution Task at PAN 2019." *CLEF (Working Notes)*, 2019.

Klaussner, Carmen. "Elements of Style Change." *University of Dublin, Ireland*, 2017.

Klaussner, Carmen, and Carl Vogel. "Stylochronometry: Timeline Prediction in Stylometric Analysis." *Research and Development in Intelligent Systems XXXII*, edited by Max Bremer and Miltos Petridis, Springer International Publishing, 2015, pp. 91–106. *Crossref*, doi:10.1007/978-3-319-25032-8_6.

---. "Temporal Predictive Regression Models for Linguistic Style Analysis." *Journal of Language Modelling*, vol. 6, no. 1, Aug. 2018, doi:10.15398/jlm.v6i1.177.

Koppel, Moshe, and Jonathan Schler. "Exploiting Stylistic Idiosyncrasies for Authorship Attribution." *Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*, vol. 69, 2003, pp. 72–80.

Lamalle, C., et al. *Lexico 3 Version 3.41 Février 03. Outils de Statistique Textuelle. Manuel d'Utilisation.* Laboratoire SYLED-CLA2T, Université de la Sorbonne nouvelle - Paris 3, 2003, http://www.lexi-co.com/ressources/manuel-3.41.pdf.

Ledger, Gerard R. *Re-Counting Plato a Computer Analysis of Plato's Style*. 1989.

Legallois, Dominique, et al. "The Balance Between Quantitative and Qualitative Literary Stylistics: How the Method of 'Motifs' Can Help." *The Grammar of Genres and Styles: From Discrete to Non-Discrete Units*, 2018, pp. 164–93.

Meyerhoff, Miriam, and James A. Walker. "The Persistence of Variation in Individual Grammars: Copula Absence in ?Urban Sojourners? And Their Stay-at-Home Peers, Bequia (St Vincent and the Grenadines)." *Journal of Sociolinguistics*, vol. 11, no. 3, June 2007, pp. 346–66. *Crossref*, doi:10.1111/j.1467-9841.2007.00327.x.

Mollin, Sandra. "'I Entirely Understand' Is a Blairism: The Methodology of Identifying Idiolectal Collocations." *International Journal of Corpus Linguistics*, vol. 14, no. 3, Aug. 2009, pp. 367–92, doi:10.1075/ijcl.14.3.04mol.

Opas, L. L. "A Multi-Dimensional Analysis of Style in Samuel Beckett's Prose Works." *Research in Humanities Computing 4.*, edited by S. Hocking and N. Ide, Clarendon Press., 1996.

Petré, Peter, et al. "Early Modern Multiloquent Authors (EMMA): Designing a Large-Scale Corpus of Individuals' Languages." *ICAME Journal*, vol. 43, no. 1, Mar. 2019, pp. 83–122, doi:10.2478/icame-2019-0004.

Petré, Peter, and Freek Van de Velde. "The Real-Time Dynamics of the Individual and the Community in Grammaticalization." *Language*, vol. 94, no. 4, 2018, pp. 867–901, doi:10.1353/lan.2018.0056.

Philippe, Gilles. *Pourquoi le style change-t-il?* Les Impressions Nouvelles, 2021, doi:10.14375/np.9782874498671.

Pincemin, Bénédicte. *Sept Logiciels de Textométrie*. 2018.

Popescu, Octavian, and Carlo Strapparava. "Semeval 2015, Task 7: Diachronic Text Evaluation." *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 2015, pp. 870–78, doi:10.18653/v1/s15-2147.

Raumolin-Brunberg, Helena, and Terttu Nevalainen. "Historical Sociolinguistics: The Corpus of Early English Correspondence." *Creating and Digitizing Language Corpora*, edited by Joan C. Beal, et al., Palgrave Macmillan UK, 2007, pp. 148–71, doi:10.1057/9780230223202_7.

Robinson, T. M. "Plato and the Computer." *Ancient Philosophy*, vol. 12, no. 2, 1992, pp. 375–82, doi:10.5840/ancientphil19921228.

Robinson, W. S. "A Method for Chronologically Ordering Archaeological Deposits." *American Antiquity*, vol. 16, no. 4, Apr. 1951, pp. 293–301, doi:10.2307/276978.

Sari, Yunita, et al. "Continuous N-Gram Representations for Authorship Attribution." *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 267–73, doi:10.18653/v1/e17-2043.

Schmidt-Petri, Christoph, et al. "Who Authored *On Liberty*? Stylometric Evidence on Harriet Taylor Mill's Contribution." *Utilitas*, vol. 34, no. 2, Dec. 2021, pp. 120–38, doi:10.1017/s0953820821000339.

Seminck, Olga, et al. "The Corpus for Idiolectal Research (CIDRE)." *Journal of Open Humanities Data*, vol. 7, 2021, p. 15, doi:10.5334/johd.42.

Smith, Joseph A., and Coleen Kelly. "Stylistic Constancy and Change across Literary Corpora: Using Measures of Lexical Richness to Date Works." *Computers and the Humanities*, vol. 36, no. 4, 2002, pp. 411–30. *Crossref*, doi:10.1023/a:1020201615753.

Stamatatos, Efstathios. "On the Robustness of Authorship Attribution Based on Character N-Gram Features." *JL & Pol'y*, vol. 21, 2012, p. 421.

Stamou, C. "Stylochronometry: Stylistic Development, Sequence of Composition, and Relative Dating." *Literary and Linguistic Computing*, vol. 23, no. 2, Oct. 2007, pp. 181–99. *Crossref*, doi:10.1093/llc/fqm029.

Sundararajan, Kalaivani, and Damon Woodard. "What Represents 'Style' in Authorship Attribution?" *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2814–22.

Temple, J. T. "A Multivariate Synthesis of Published Platonic Stylometric Data." *Literary and Linguistic Computing*, vol. 11, no. 2, June 1996, pp. 67–75. *Crossref*, doi:10.1093/llc/11.2.67.

Thibaudet, Albert. *Gustave Flaubert*. Éditions Gallimard, 1922.

Vanni, Laurent, et al. "Hyperdeep: Deep Learning Descriptif Pour l'analyse de Données Textuelles." *JADT 2020*, 2020.

Whissell, Cynthia. "Traditional and Emotional Stylometric Analysis of the Songs of Beatles Paul McCartney and John Lennon." *Computers and the Humanities*, vol. 30, no. 3, 1996, pp. 257–65, doi:10.1007/bf00055109.

Wishart, David, and Stephen V. Leach. "A Multivariate Analysis of Platonic Prose Rhythm." *Computer Studies in the Humanities and Verbal Behavior*, vol. 3, no. 2, 1970, pp. 90–99.

XML, BNC. *The British National Corpus XML Edition DVD*. Oxford: Oxford University Press, 2007, http://www.natcorp.ox.ac.uk/docs/URG/.

Zhang, Richong, et al. "Syntax Encoding with Application in Authorship Attribution." *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2742–53, doi:10.18653/v1/d18-1294.

On Distances between Words with Parameters

Pierre Bourhis, Aaron Boussidan, Philippe Gambette

On Distances between Words with Parameters

Pierre Bourhis
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

Aaron Boussidan
LIGM, Université Gustave Eiffel, CNRS, France

Philippe Gambette
LIGM, Université Gustave Eiffel, CNRS, France

Abstract

The edit distance between parameterized words is a generalization of the classical edit distance where it is allowed to map particular letters of the first word, called parameters, to parameters of the second word before computing the distance. This problem has been introduced in particular for detection of code duplication, and the notion of words with parameters has also been used with different semantics in other fields. The complexity of several variants of edit distances between parameterized words has been studied, however, the complexity of the most natural one, the Levenshtein distance, remained open.

In this paper, we solve this open question and close the exhaustive analysis of all cases of parameterized word matching and function matching, showing that these problems are NP-complete. To this aim, we also provide a comparison of the different problems, exhibiting several equivalences between them. We also provide and implement a MaxSAT encoding of the problem, as well as a simple FPT algorithm in the alphabet size, and study their efficiency on real data in the context of theater play structure comparison.

2012 ACM Subject Classification Theory of computation → Pattern matching

Keywords and phrases String matching, edit distance, Levenshtein, parameterized matching, parameterized words, parameter words, instantiable words, NP-completeness, MAX-SAT.

Digital Object Identifier 10.4230/LIPIcs.CPM.2023.16

34th Annual Symposium on Combinatorial Pattern Matching (CPM 2023).
Editors: Laurent Bulteau and Zsuzsanna Lipták; Article No. 16; pp. 16:1–16:24

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Measuring the similarity between text strings is a fundamental problem in computer science, and has applications in bioinformatics [23], databases [1, 16] and natural language processing [27]. Among the measures of similarities between strings, the Levenshtein distance [28] is the most commonly used, both for its practicality and its ease of computation. This distance quantifies the minimum number of operations of insertion, deletion, and substitution needed to transform a string into another one. It has a wide range of applications, ranging from biological sequence alignment [33] to dialect pronunciation differences [25] or signature authentication [34]. Computing the edit distance between two strings of length n and m can be achieved in time $O(nm)$, by computing the distance between all their prefixes, and storing the results in a dynamic programming fashion [37]. The success of the Levenshtein distance generated many extensions and generalization on more complex models, such as trees [38] or automata [32].

However, a limitation of the Levenshtein distance is that it only captures proximity between strings (or objects) written on the same alphabet. Evaluating the proximity of strings written on different alphabets is a problem that arises in various applications, such as bioinformatics [35], image processing [17] and code duplication [6, 7]. In all those contexts, the technique used is the one of parameterized matching [6, 7]. Instead of using classical strings, parameterized matching uses "parameterized words" written using both constant parts, which are expensive to rename, and parameters, which are meant to be renamed freely. Formally, two equal-length strings u and v over an alphabet Π are said to be parameterized matching if there exists a 1-to-1 function $f : \Pi \to \Pi$ such that $f(u) = v$, where $f(u)$ is defined as $f(u_1) \dots f(u_{|u|})$.

Words with parameters also occur in other frameworks, and are often used in slightly different ways. The first of those frameworks was initially introduced in the context of Ramsey theory in the 80s [36], and is called "parameter words". In this context, parameters are labelled according to their order of first occurrence. Parameter words are also equipped with a composition operation, where parameters of the first word can be instantiated by characters or parameters of the second word. Parameter words can also be seen as equivalence classes of parameterized words, which are the main focus of this article.

A second framework using parameters is the one of parameterized regular expressions introduced in [10], where parameters can only be instantiated by constants, and not by other parameters. The focus in this context is therefore made on the set of all possible valuations of the parameters. Then, when defining algorithmic problems on such objects, two distinct semantics can be studied: either the "certainty semantics", where all valuations need to have some property, or the "possibility semantics", where at least one valuation needs to have this property. To make a difference with the parameterized word framework mentioned below, we choose to call these words "instantiable words". Finally, this notion of words with parameters can also be seen as a refined version of partial words (words containing a wildcard character) [15]. The notion of partial words is also important in the context of databases where paths of incomplete graphs can be interpreted as instantiable words [9].

This article aims at studying similarity by using edit distances in the framework of words with parameters. In this framework, the pattern matching problem, which consists in looking for the first string as a subword of the second string, has been extensively studied, either looking for exact occurrences, with efficient algorithms [4, 19, 30] or approximate ones, which is often NP-hard [21, 22]. In the case where we compare the two input strings in their entirety, various exact parameterized matching problems have been studied for parameterized

d	$\emptyset$	D	I	DI
$\emptyset$	P [8]	NP-complete (Th. 12)	NP-complete (Cor. 14)	NP-complete [26]
S	P [24]	NP-complete (Cor. 14)	NP-complete (Cor. 14)	NP-complete (Th. 13)

Figure 1 Complexity of the variants of parameterized matching PM^d, depending on the kind of operations (D: deletion, I: insertion, S: substitution) allowed in the edit distance d.

pattern matching, namely string parameterized matching [7], single pattern parameterized matching [7, 3], multiple pattern parameterized matching, or 2-dimensional parameterized matching, many of those works being compiled in [29] and [31]. Different approximate variants of parameterized matching using edit distance have already been studied, but the problem has not been completely solved: the first work on the topic is [8], in which Baker introduces a form of approximate parameterized pattern matching in which the replacement of any substring by another one that is in parameterized matching with it is considered as a base edit operation. Parameterized matching under the Hamming distance, i.e., with a distance allowing only substitutions, has been covered in [24], where the authors prove that the string matching problem with at most k mismatches can be solved in time $O(m + k^{1.5})$. The LCPS (Longest Common Parameterized Subsequence) problem, equivalent to the parameterized pattern matching problem with insertions and deletions, is shown to be NP-hard in [26], which also provides an approximation algorithm. Those two different complexity classes for these problems raise the question of the complexity of the problem under the Levenshtein distance. This problem was left as an open question in the conclusion of [24].

Our paper establishes that this problem is NP-complete. Moreover, the result also extends to any possible edit distances obtained from deletion, insertion, and substitution as soon as substitution is not the only operation allowed, as summarized in Figure 1. Our main proof also implies the main theorem of [26] with a different NP-completeness reduction. This contrasts with the problems of exact parameterized pattern matching which are all polynomial-time solvable, as well as all variants of the string matching problem with deletions, insertions or substitutions.

We also extend these results to function matching, which is the problem obtained by relaxing the 1-to-1 restriction in parameterized matching, as defined in [2]. This generalization, by breaking the symmetry of parameterized matching, actually gives rise to two close but different problems, depending of the order of operations that are considered. We study the links between all these problems and their computational complexity, and study two practical ways to solve them, parameterized complexity and the use of maxSAT solvers.

We also make a direct connection with the framework of instantiable words, more precisely with a natural problem of distance between languages. We show how instantiable word problems can be reduced to parameterized matching ones, under the right assumptions. This allows us to open new perspectives on the complexity of several language repair problems.

In Section 2, we give basic definitions and notations, and recall the existing formalism of parameterized matching and instantiable words. In Section 3 we discuss approximate parameterized matching and its various generalizations. We also link it to instantiable words. In Section 4, we first prove a collection of technical results that build up to the NP-completeness proofs for parameterized matching and function matching problems defined above. In Section 5, we study two approaches to solve one of the variants of parameterized matching in practice, a simple FPT algorithm parameterized by the alphabet size and a MaxSAT encoding. We show in Section 6 that these implementations can solve real instances of the problem, motivated by structure comparison of theater plays.

Finally, in Section 7, we conclude the paper and give a few perspectives on the notion of distance between parameterized languages.

2 Notations and Definitions

2.1 Basic Notations on Words and Editions

Words

An **alphabet** is a set of letters. A word on an alphabet A is a finite sequence of letters from A, indexed starting from 1. Let u be a word on A. Unless defined differently, we note u_i the i-th letter of u, and $|u|$ is the length of u. If $i \notin [1, |u|]$, u_i is defined as the empty word ε. If x is a letter from A, $|u|_x$ is the number of times x appears in u. Similarly, if X is a set of letters, $|u|_X = \sum_{x \in X} |u|_x$ is the number of occurrences of letters of X in u. If f is a function defined on an alphabet A, we extend it to A^* in the usual way, so that $f(u) = f(u_1) \dots f(u_{|u|})$. If f is a function, we denote by $\mathcal{D}(f)$ the domain of f. Two functions f and g are said to be **compatible** if $f|_{\mathcal{D}(g)\cap\mathcal{D}(f)} = g|_{\mathcal{D}(g)\cap\mathcal{D}(f)}$. The identity function on D is defined as $Id_D(x) = x$ for all x in D.

Edit Operations

In this paper, we consider the three classical **edit operations** which are *deletion*, *substitution* and *insertion*. Let $u = u_1 \dots u_n$ be a word of size n. Let i be an integer between 0 and n and x be a letter of the alphabet, the **insertion** at position i is the operation that transforms u to $u_1 \dots u_i x u_{i+1} \dots u_n$ Let j be an integer between 1 and n, the **deletion** at position j is the operation that transforms u into $u_1 \dots u_{j-1} u_{j+1} \dots u_n$. Let y be a letter of the alphabet and $y \neq u_j$, the **substitution** to y at position j is the operation that transforms u into $u_1 \dots u_{j-1} y u_{j+1} \dots u_n$. A **sequence of operations** or **rewriting sequence** ρ is a sequence of edit operations. We denote by $\rho(u)$ the word obtained by applying the edit operations of ρ one after another, in the order defined by ρ, to u.

Distances

Given a set of edit operations E and two words u and v, the **edit distance** between u and v is defined as the length of a shortest sequence of operations of E changing u into v. We denote by D the distance obtained on words by allowing only deletion operations: that is to say $D(u, v) = k$ iff v can be obtained by deleting k letters from u. Similarly, we note I and S the distances obtained by allowing only insertions and substitutions respectively (note that S is the Hamming distance). We also combine these notations to define DI as the distance with insertions and deletions, and so on. We also denote the Levenshtein distance DIS by L. Note that some of these edit distances are not metrics, because they are not symmetrical. We emphasize this by calling **symmetric edit distances** the distances DI, S, and L.

2.2 Comparing Words with Parameters

Conceptually, a word with parameters is a word in which some letters are not yet determined. In order to distinguish the parameters from the constants, we split the alphabet into Σ, the alphabet of the constants and Π, the alphabet of the parameters. By definition, these alphabets are finite. A word with parameters can either be seen as representing a "word template" (i.e., a word with variable parts), or a set of words (determined by all possible

affectations of its parameters). Depending on the definition chosen, comparing two words w_1 and w_2 is done in two different ways. In the first setting [6, 7, 8, 31, 2, 5, 24, 29, 26, 17], parameters of w_1 are renamed through a function f that maps the set of parameters to itself, and acts as identity on the set of constants. It is then possible to compare $f(w_1)$ and w_2, which are written on the same alphabet. In the second setting, constants are seen as the concrete values parameters can take [11]. Parameters are instantiated through two functions f_1 and f_2 that map constants to themselves, but also map parameters to constants. The words $f_1(w_1)$ and $f_2(w_2)$ are then made only of constants, and can be compared. Formally, this gives rise to the two following different definitions:

On the one hand, a **parameterized word** is a word on an alphabet $\Sigma \cup \Pi$. In all that follows, Σ and Π are two disjoint alphabets, respectively called the **alphabet of constants** and the **alphabet of parameters**. Alphabets Σ and Π are considered to be finite, unless specified otherwise.

Two parameterized words u and v are said to be in **function matching** if there exist $f_\Pi : \Pi \to \Pi$ and $f : \Pi \cup \Sigma \to \Pi \cup \Sigma$ such that $f|_\Pi = f_\Pi$, $f|_\Sigma = Id_\Sigma$, and $f(u) = v$. In the classical setting [6], f is also **constrained to be 1-to-1**, and this relationship is called **parameterized matching**. Note that parameterized matching is an equivalence relation on parameterized words. Testing if two words are parameterized matching can be achieved in linear time [7].

On the other hand, an **instantiable word** is a word on the alphabet $\Sigma \cup \Pi$. Given $f : \Pi \to \Sigma$, we extend it to constants by setting $f(x) = x$ for all $x \in \Sigma$, and we then define the **language of an instantiable word** u to be $L(u) = \{w \in \Sigma^* \mid \exists f : \Pi \to \Sigma, f(u) = w\}$. This definition is akin to the $L_\diamond$ semantic of a parameterized regular expression defined in [11], but restricted here to a single instantiable word. Two instantiable words w_1 and w_2 describe the same elements if their languages are equal, i.e. $L(w_1) = L(w_2)$.

3 Different Definitions for Different Semantics and Problems

In this section, we introduce various new approximate variants of parameterized matching, and compare them, highlighting their differences on examples.

3.1 Variants of Parameterized Matching

In parameterized matching, the function f renaming parameters is generally considered to be 1-to-1. In this paper, we also consider the **function matching problem**, which is the case where f is not constrained to be injective anymore, as defined in [2]. We also introduce multiple approximate variants of the parameterized matching problems, depending on several edit distances obtained by combining insertion, deletion and substitution operations.

3.1.1 Edit distances for parameterized matching between two strings: PM^d

▶ **Definition 1.** *If d is an edit distance, we denote by PM^d the parameterized matchingproblem under d, which is the following:*

- ***Input:*** *two parameterized words u, v, a parameter alphabet Π , an alphabet Σ of constants, and a natural number k.*
- ***Problem:*** *Does there exist u' such that $d(u, u') \leq k$ and u' and v are parameterized matching, i.e. there exists a 1-to-1 function $f : \Pi \cup \Sigma \to \Pi \cup \Sigma$ such that $f|_\Sigma = Id_\Sigma$, $f(\Pi) = \Pi$, and $f(u') = v$?*

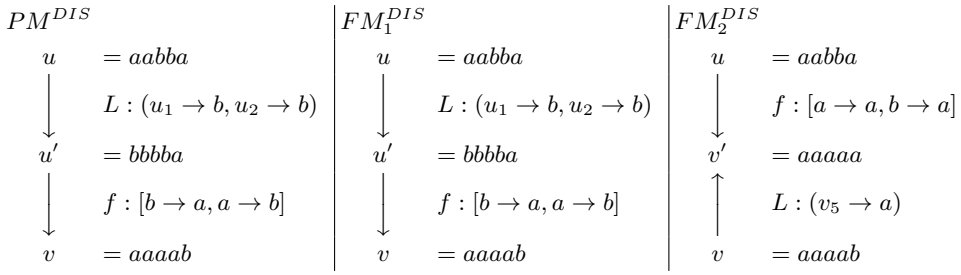


Figure 2 Side-by-side comparison of PM^{DIS}, FM_1^{DIS} and FM_2^{DIS}

In that case, we say that u' and f **realize** the matching between u and v. We sometimes write that only f or u' realize the matching if the other one is not relevant to a proof.

In cases where Σ and Π are already defined, we omit them and simply call $PM^d(u, v, k)$ the result of the decision problem. Furthermore, $PM^d(u, v)$ denotes the minimum integer k (potentially infinite) such that $PM^d(u, v, k)$ is true.

We can note that this problem can be solved in polynomial time adapting the classical dynamic programming algorithm [33, 37] when the alphabet sizes are fixed.

3.1.2 Edit distances for function matching between 2 strings: FM_i^d

To denote **function matching problems**, we use FM instead of PM: FM^D denotes the function matching problems with deletions.

Furthermore, if $\mathcal{P}$ is one of the problems defined above, we note $\mathcal{P}_1$ the problem where edit operations are only applied to the first argument, and $\mathcal{P}_2$ the one where they are only applied to the second argument.

▶ **Definition 2.** *The FM_1^d and FM_2^d problems are defined as follows. For both problems, the input is the following:*

- ***Input:*** *two parameterized words u, v, a parameter alphabet Π, a constant alphabet Σ, and a natural number k.*

The problems are then:

- ***Problem FM_1^d:*** *$\exists u'$ such that $d(u, u') \leq k$ and u' and v are function matching?*
- ***Problem FM_2^d:*** *$\exists v'$ such that $d(v, v') \leq k$ and u and v' are in function matching?*

Note that the renaming function f is always applied to one input only. These definitions are illustrated on an example in Figure 2.

3.2 Comparing Variants of PM

In this subsection, we compare the different variants of our problem.

Regarding the one-to-one parameterized matching PM, note that the definition we give above is designed to be easily extended to the different variants when we drop the one-to-one restriction. In [24], the authors consider that the "correct way for defining the edit distance problem" is "to allow the operations and then apply the edit distance". By extending the definition of FM_1^d and FM_2^d to define PM_1^d and PM_2^d in the case of one-to-one matching, we see that it is actually possible to switch the order of operations, and to reverse them (deletions then become insertions and vice versa, and the renaming function f^{-1} is well-defined), in this case. This makes our definition consistent with the quote from [24]

above. Formally, this gives the following equalities, for all parameterized words u and v: $PM_1^I(u,v) = PM_1^D(v,u) = PM_2^D(u,v) = PM_2^I(v,u)$.

More generally, it holds that for every edit distance d, $PM_1^d(u,v) = PM_1^{d^{-1}}(v,u) = PM_2^{d^{-1}}(u,v) = PM_2^d(v,u)$, where d^{-1} denotes the converse distance of d, i.e. d^{-1} contains deletions if d contains insertions, insertions if d contains deletions, and substitutions if d contains substitutions.

However, for function matching, we only have the following equalities: $FM_1^I(u,v) = FM_2^D(u,v)$ and $FM_1^D(u,v) = FM_2^I(u,v)$.

By taking $u = ab$ and $v = cc$, we can notice that $FM_1^I(u,v) = 0$ and $FM_1^D(v,u) = \infty$, so the equality $FM_1^I(u,v) = FM_1^D(v,u)$ does not hold.

Finally, note the following inequalities:

▶ **Proposition 3.** *Let u and v be parameterized words over $\Sigma \cup \Pi$. Then:*

1. $FM_1^d(u,v) \leq PM^d(u,v)$;
2. *If d is a symmetric edit distance, $FM_2^d(u,v) \leq FM_1^d(u,v)$.*

Proof. The first point comes from the fact that any solution to PM^d is also a solution to FM_1^d. For the second point, let $k = FM_1^d(u,v)$, and let u' and f realize $FM_1^d(u,v)$. We construct a word v', obtained by applying to v the same operations applied to u to obtain u', but "mirrored". That is to say, for every operation used in u, we apply an operation in v, in the following way:

- If a letter a is inserted in u, there exists a position i in u' such that $u'_i = a$, and $f(u'_i) = v_i$. Hence, we delete v_i in v.
- Similarly, if a letter is substituted for another letter a' in u, there exists i such that $u'_i = a$, and we substitute v_i to $f(a)$.
- If a letter a is deleted in u at position i, we insert $f(a)$ in v at position i instead.

It then holds that $f(u) = v'$, and hence $PM_2^d(u,v) \leq k$. ◀

Note that the above proof does not work to prove the converse inequality between FM_1^d and FM_2^d, as it would require to consider an element of $f^{-1}(a)$, which might be empty. This is illustrated in the following example, on the alphabet $\Pi = \{a, b\}$:

▶ **Example 4.** Let $N \in \mathbb{N}$ and consider $u = a^N b^N b$ and $v = a^N a^N b$. u and v are not in parameterized matching, hence $FM_1^{DIS}(u,v) > 0$ and $FM_2^{DIS}(u,v) > 0$. By substituting the last b in v for a a, and picking a function f such that $f(a) = f(b) = a$, we get $FM_2^{DIS}(u,v) = 1$ (see Figure 2 for an example with $N = 2$). For FM_1^{DIS}, since b appears in v, it holds that for any function f realizing FM_1^{DIS}, $f(a) = b$ or $f(b) = b$. Hence, at least N occurrences of b appear in $f(u)$. Since there is only one occurrence of b in v, it is clear that $FM_1^{DIS}(u,v) \geq N - 1$.

The difference between FM_1^d and FM_2^d comes from the fact that Π is fixed in the input. In the case where Π could be extended, both problems can be shown equivalent (for example if we allow a new letter c in the example of Figure 2, we also get $FM_1^{DIS}(u,v) = 1$ by setting $u_5 \to c$ and $f : [a \to a, b \to a, c \to b]$), by using the same proof as Proposition 3.

3.3 Instantiable Words versus Parameterized Words

The parameterized word formalism and the instantiable word formalism give rise to two different definitions of distances between words. Given an edit distance d on words, there are two ways to extend it to words with parameters. Let w_1 and w_2 be two words over $\Sigma \cup \Pi$. The two possible extensions are the following:

- The distance between w_1 and w_2 is defined as $d(w_1, w_2) = PM^d(w_1, w_2)$. Alternatively, the function distance between w_1 and w_2 is defined as $FM_1^d(w_1, w_2)$.
- The distance between w_1 and w_2 is the distance between their respective languages seen as sets, that is to say $d(w_1, w_2) = d(L(w_1), L(w_2)) = \sup_{u \in L(w_1)} \inf_{v \in L(w_2)} d(u, v)$. Equivalently, $d(w_1, w_2) \leq k$ if and only if for all $f_1 : \Pi \to \Sigma$, there exists $f_2 : \Pi \to \Sigma$ such that $d(f_1(w_1), f_2(w_2)) \leq k$.

This second definition stems from the definition of distance between languages, as defined and studied in [12, 13, 14].

▶ **Example 5.** Consider the words $u = axyb$ and $v = xbby$, on the alphabets $\Sigma = \{a, b\}$ and $\Pi = \{x, y\}$, and consider the distance S. On the one hand, $FM_1^S(u, v) = 4$, because regardless of the matching chosen, every letter of $f(u)$ has to be substituted. On the other hand, for any function $f_1 : \Pi \to \Sigma$, choosing f_2 such that $f_2(x) = a$ and $f_2(y) = b$ yields a distance $d(f_1(u), f_2(v))$ of at most 2, by substituting the 2 middle letters.

Given a big enough alphabet, those two definitions can in fact be shown equivalent:

▶ **Proposition 6.** *Let w_1 and w_2 be words over $\Sigma \cup \Pi$, and let d be a symmetric edit distance on $\Sigma \cup \Pi$. Suppose $|\Sigma| \geq |w_1| + |w_2|$, and let k be an integer. Then, the following are equivalent:*

1. $FM_1^d(w_2, w_1) \leq k$
2. $d(L(w_1), L(w_2)) \leq k$

Notice how w_1 and w_2 change position between the two distances. This is not benign, as FM_1^d is not symmetric.

Proof. Suppose $FM_1^d(w_2, w_1) \leq k$. There exists $f : \Pi \to \Pi$ such that $d(f(w_2), w_1) \leq k$. For this proof, we will use the characterization of the distance betweeen languages with f_1 and f_2. Let $f_1 : \Pi \to \Sigma$. Define $f_2 = f_1 \circ f$. Since $d(w_1, f(w_2)) \leq k$, we have $d(f_1(w_1), f_1 \circ f(w_2)) \leq k$, by following the same edit operations. Hence $d(f_1(w_1), f_2(w_2)) \leq k$.

Suppose now $d(L(w_1), L(w_2)) \leq k$. Let $f_1 : \Pi \to \Sigma$ be a 1-to-1 function such that for all parameters x in w_1, $f(x)$ does not appear in w_1 or w_2. This is possible since Σ is large enough. There exists $f_2 : \Pi \to \Sigma$ such that $d(f_1(w_1), f_2(w_2)) \leq k$. Let $h : \Sigma \to \Pi \cup \Sigma$ be such that if $x \in \Pi$, $h(f_1(x)) = x$, and if $x \notin f_1(\Pi)$, $h(x) = x$. We then have $h \circ f_1 = Id$. What is more, since h is injective, $d(f_1(w_1), f_2(w_2)) = d(h(f_1(w_1)), h(f_2(w_2)) = d(h(f_2(w_2)), w_1)$. Hence, $FM_1^d(w_2, w_1) \leq k$. ◀

4 Hardness Results for Approximate Parameterized Matching

In this section, we study the complexity of the various parameterized matching problems. We show the NP-completeness of the simplest problems using only deletions, which will be sufficient to show the NP-completeness of all the other problems. We start by proving some practical lemmas, and then proceed to the reductions.

4.1 "Block by block" Lemmas

In this section, we regroup a few useful technical lemmas. We start of by stating two simple results on distance and words, for which the proofs can be found in Appendix A. We then turn to block lemmas, which will later be useful in the proofs of Theorems 12,17 and 15, to combine the various gadgets defined during the reduction.

This lemma simply states a commutativity result between the application of a matching f and the rewriting steps.

▶ **Lemma 7.** *Let d be a distance, k an integer and u, v two parameterized words such that $PM^d(u,v) \leq k$, and let f realize this parameterized match. Then: $d(f(u), v) \leq k$. The same result holds for $FM_1^d(u,v)$.*

Proof idea: The proof is done by induction on k. We discuss whether the $(k+1)$-th operation is an insertion, a deletion, or a substitution, and show that a corresponding operation can be used in $f(u)$.

▶ **Lemma 8.** *Let z, u and v be (parameterized) words, and let d be a distance. Then $d(zu, zv) = d(u, v)$.*

Proof Idea: We show that we can consider every rewriting operation to be applied in u only: if z is modified during an optimal rewriting sequence, the words have some redundancy, and the same operations could have been carried in u instead. We proceed again by induction, and focus on the base case by studying the 3 possible cases, one for each type of operation.

Next, we turn to prove "block by block" matching lemmas. Those results state that it is possible to encode multiple parameterized matching instances into a single one. They hold for every type of problems considered here, but their proofs vary slightly; we present them in order of increasing complexity. Note that all the constructions given can be achieved in polynomial time.

▶ **Lemma 9.** *Let $u_1, \dots u_n$ and $v_1, \dots v_n$ be parameterized words over $\Sigma \cup \Pi$ such that for $1 \leq i \leq n$, $k_i = |u_i| - |v_i| \geq 0$, and $k = \sum_{i=1}^{n} k_i$. There exist u and v two parameterized words over $\{\#\} \cup \Sigma \cup \Pi$, where $\#$ is a fresh variable, such that the following are equivalent:*

1. *$PM^D(u,v) = k$*
2. *For all $1 \leq i \leq n$, $PM^D(u_i, v_i) = k_i$ and the applications f_i realizing those matchings are all compatible.*

Proof. The idea behind this proof and all the following ones is that we can introduce a separator $\#$ to concatenate all the words, and that this separator will never be touched by any deletions or applications of f.

Let $\#$ be a fresh constant. We define $u = u_1\#u_2\#\dots\#u_n$, and $v = v_1\#v_2\#\dots\#v_n$.

2. $\implies$ 1.: For every $1 \leq i \leq n$, take u'_i and f_i to realize the matchings. We can obtain $u' = u'_1\#u'_2\dots\#u'_n$ from u by applying the same deletions. Taking f to be the smallest function extending all the f_i, we get $PM^D(u,v) \leq k$.

1. $\implies$ 2.: Assume $PM^D(u,v) \leq k$. Let u' and f realize this parameterized match. Since the $\#$ symbols are constants, we have $f(\#) = \#$. Since u' is obtained from u by deletions, we have $|u'|_\# \leq |u|_\#$. Since f is injective and $f(\#) = \#$, $|f(u')|_\# \leq |f(u)|_\#$. Hence, it holds that $|v|_\# = |f(u')|_\# \leq |f(u)|_\# = |u|_\#$. Since $|u|_\# = |v|_\#$, this is an equality, and $|f(u')|_\# = |f(u)|_\#$. Hence $|u'|_\# = |u|_\#$, and no $\#$ character is deleted. The word u' is then of the form $u'_1\#u'_2\#\dots\#u'_n$, where $|u'_i|_\# = 0$ and $D(u_i, u'_i) = k_i$ for all i. Thus, $f(u') = f(u'_1)\#f(u'_2)\#\dots\#f(u'_n) = v_1\#v_2\#\dots_\# v_n$. Since no other $\#$ letter appear in any $f(u'_i)$ and v_i, we can deduce that $f(u'_i) = v_i$ for all i. Finally, this yields $PM^D(u_i, v_i) = k$, and taking all the $f_i = f$ gives all compatible functions, which concludes the proof. ◀

In this proof, we used a constant $\#$. However, it can also be conducted without using a constant alphabet; indeed, constants can be encoded with parameters, as shown in Appendix B.

Lemma 9 is still valid if PM^D is replaced by FM_2^D. This time, we conduct this proof without resorting to the use of constants. This result will be used twice : once for the proof

of theorem 17, and again to show that we can once more encode constants into Π using Lemma 25 in Appendix B.

▶ **Lemma 10.** *Let $u_1, \dots u_n$ and $v_1, \dots v_n$ be parameterized words over Π such that $k_i = |v_i| - |u_i| \geq 0$, and $k = \sum_{i=1}^{n} k_i$. Then there exist u and v, two parameterized words over $\Pi \cup \{\#\}$, where $\#$ is a fresh variable, such that the following are equivalent:*

1. *$FM_2^D(u, v) \leq k$*
2. *For all $1 \leq i \leq n$, $FM_2^D(u_i, v_i) \leq k_i$, and the applications f_i realizing those matchings are all compatible.*

Proof idea: The same technique as Lemma 9 is used but u and v are defined as $u = \#^{k+1}u_1\#u_2\# \dots \#u_n$ and $v = \#^{k+1}v_1\#v_2\# \dots \#v_n$ where $\#^{k+1}$ denotes $k+1$ repetitions of the character $\#$. The full proof can be found in Appendix A.

Finally, the same block result holds for FM_1^D, and will be used in the proof of theorem 15

▶ **Lemma 11.** *Let $u_1, \dots u_n$ and $v_1, \dots v_n$ be parameterized words over Π such that for every $1 \leq i \leq n$, $k_i = |u_i| - |v_i| \geq 0$, and $k = \sum_{i=1}^{n} k_i$. Then there exist u and v two parameterized words over $\Pi \cup \{\#\}$, where $\#$ is a fresh variable, such that the following are equivalent:*

1. *$FM_1^D(u, v) \leq k$*
2. *For all $1 \leq i \leq n$, $FM_1^D(u_i, v_i) \leq k_i$, and the applications f_i realizing those matchings are all compatible.*

Proof idea: The difference with Lemma 10 is that some $\#$ symbols might be deleted, while some base letters could be mapped to $\#$. To ensure this does not happen, we define $u = \#^N u_1 \#^N u_2 \dots \#^N u_n \#^N$ and $v = \#^N v_1 \#^N v_2 \dots \#^N v_n \#^N$. The full proof can be found in Appendix A.

The technique of block-by-block matching will be used in all the reductions, to encode multiple constraints in a single PM or FM instance.

4.2 1-to-1 Parameterized Matching PM

We now focus on the complexity of the PM^d problems. These problems, as well as function matching problems, are all clearly in NP: given the list of deletion, insertion or substitution operations to do and the matching to apply, it is easy to check that the solution is correct.

For the reductions in this paper, we always assume that words are written without constants, that is to say $\Sigma = \emptyset$, since this is sufficient for NP-completeness results. This choice is also motivated by the results of Appendix B, which show that Σ can in most cases be coded into Π.

▶ **Theorem 12.** *The 1-to-1 Parameterized Matching with deletions PM^D is NP-complete.*

The proof is a reduction from the NP-complete problem **3-coloring**[20]. Given an instance G of **3-coloring**, we will construct two words u and v. The word v will represent the list of vertices and edges of G, while the word u will list the color of each vertex, and the possible coloring of each pair of vertices joined by an edge. By deleting characters from u, we make a choice for the coloring of each vertex, and thus each edge. The function f answering the parameterized matching problem will assign a choice of color to each vertex. The instance that we define should be positive iff G is 3-colorable. More formally:

Proof. The 3-Coloring problem is defined as follows :

- **Input:** $G = (V, E)$ a graph with vertices V and edges E
- **Output:** A coloring $c : V \to \{c_1, c_2, c_3\}$ such that for all $\{u, v\} \in E$, $c(u) \neq c(v)$

Let $G = (V, E)$ be an instance of **3-Coloring**, and let $V = \{x_1, \dots, x_n\}$ be the set of its n vertices, and $E = \{e_1, \dots, e_m\}$ be the set of its edges. The parameter alphabet Π, of polynomial size in $O(|G|)$ will contain:

- $x_1, \dots x_n$, corresponding to the vertices of G;
- n copies of the parameters corresponding to the colors c_1, c_2 and c_3: c_1^i, c_2^i, c_3^i for $1 \leq i \leq n$;
- for every edge e, the delimiters Y^e and $\square_1^e, \dots \square_6^e$;
- $2n$ bottom symbols, $\bot_1^i$, $\bot_2^i$ for $1 \leq i \leq n$, which will be used to fix the image of some parameters.

First, we define words that will encode the constraint that each vertex is colored, and we make sure that the unused color variables cannot be assigned elsewhere. For $1 \leq i \leq n$, $u_1^i = u_\bot^i = c_1^i c_2^i c_3^i$, $v_1^i = x_i$ and $v_\bot^i = \bot_1^i \bot_2^i$. We then define words that include all possible colorings of each edge, and we make sure to use enough brackets. For every edge $e = \{x_i, x_j\}$, we define $u_2^e = \square_1^e c_1^i c_2^j \square_1^e \ \square_2^e c_1^i c_3^j \square_2^e \ \square_3^e c_2^i c_1^j \square_3^e \ \square_4^e c_2^i c_3^j \square_4^e \ \square_5^e c_3^i c_1^j \square_5^e \ \square_6^e c_3^i c_2^j \square_6^e$ and $v_2^e = Y^e x_i x_j Y^e$.

Applying Lemma 9 to $u_1^1, \dots u_1^n, u_\bot^1, \dots u_\bot^n, u_2^{e_1}, \dots u_2^{e_m}$ and $v_1^1 \dots v_1^n, v_\bot^1, \dots v_\bot^n, v_2^{e_1}, \dots v_2^{e_m}$, we obtain u and v. Let $k = |u| - |v| = 3n + 20m$. We now show that G is 3-colorable $\Leftrightarrow$ $PM^D(u, v) \leq k$.

$\Rightarrow$: Suppose G is 3-colorable. Let $c : V \to \{c_1, c_2, c_3\}$ be a 3-coloring of G. We define f in the following way, for $1 \leq y \leq 3$:

$$f(c_y^i) = \begin{cases} x_i & \text{if } c(x_i) = c_y, \\ \bot_1^i & \text{if } y \text{ is the smallest integer in } \{1,2,3\} \text{ such that } c(x_i) \neq c_y, \\ \bot_2^i & \text{otherwise.} \end{cases}$$

For every edge $e = \{x_i, x_j\} \in E$, since c is a valid coloring, and since every allowed arrangements of the colors is in u_2^e, there exists a unique factor of the form $\square_y^e f^{-1}(x_i) f^{-1}(x_j) \square_y^e$ in u_2^e, for some $1 \leq y \leq n$. Hence, we define $f(\square_y^e) = Y^e$. The function f can then be extended in any way to be 1-to-1 (the remaining characters whose image under f are not yet defined will all be deleted in what follows, so their image doesn't matter).

By using f defined in this way:

- For $1 \leq i \leq n$, $PM^D(u_1^i, v_1^i) \leq 2$, by deleting the 2 colors not matching the color of x_i;
- For $1 \leq i \leq n$, $PM^D(u_\bot^i, v_\bot^i) \leq 1$;
- For every edge $e \in E$, $PM^D(u_2^e, v_2^e) \leq 20$, by keeping only the factor delimited by the $\square_y^e$ symbols defined above.

Thus Lemma 9 yields $PM^D(u, v) \leq k$.

$\Leftarrow$: We now suppose u and v are a parameterized match with k deletions. The following can then be derived about f:

1. Since the u_1^i and v_1^i are matching for $1 \leq i \leq n$, there exists an element $c \in \{c_1^i, c_2^i, c_3^i\}$ such that $f(c) = x_i$. Each of these matchings is done with exactly 2 deletions, for a total of $2n$.
2. Since the $u_\bot^i$ and $v_\bot^i$ are in matching, the two other colors that are not sent to x_i are sent to $\bot_1^i$ and $\bot_2^i$. Each of these matchings is done with exactly one deletion, for a total of n.
3. For every edge $e \in E$, u_2^e and v_2^e are in matching. Let $u_2^{e\prime}$ realize this matching. For every $1 \leq i \leq n$ and $1 \leq i' \leq 3$ the colors $c_{i'}^i$ have images that are different from Y^e, so there necessarily exists $1 \leq y \leq 6$ such that $f(\square_y^e) = Y^e$. Furthermore, since f is injective, $|v_2^e|_{Y^e} = |u_2^{e\prime}|_{\square_y^e}$. Since $|v_2^e|_{Y^e} = |u_2^e|_{\square_y^e} = 2$, no $\square_y^e$ is deleted from u. Since there are two

characters between the Y^e in v_2^e and none outside, $u_2^{e\prime}$ has the same structure, and all other $\square_{y'}^e$ for $y' \neq y$ and all other colors are deleted from u_2^e.

Finally, $u_2^{e\prime}$ is of the form $\square_y^e c_t c_{t'} \square_y^e$, where $t \neq t'$ are elements of $\{1, 2, 3\}$. Each of these matchings is done with exactly 20 deletions, for a total of $20m$.

The function f then implies a coloring of G. Formally, we define $col(c_y^i) = c_y$ for $1 \leq i \leq n$ and $1 \leq y \leq 3$. We can then define $c : V \to \{c_1, c_2, c_3\}$ such that $c(x_i) = col(f^{-1}(x_i))$. Point 1 above ensures that this function definition is correct. Furthermore, for every edge $e = \{x_i, x_j\}$, point 3 ensures that $c(x_i) \neq c(x_j)$, and thus c is a valid coloring of G. ◀

This first NP-completeness results yields a few immediate corollary results, and in particular, the NP-completeness of the problem under the Levenshtein distance:

▶ **Theorem 13.** *The problem PM^{DIS} of parameterized matching under the Levenshtein distance is* NP*-complete.*

Proof. We do a simple reduction from PM^D. Let u, v, k be an instance of PM^D. If the instance is trivially false (that is to say, $k \neq |u| - |v|$), answer negatively. Else, consider u, v, k as an instance of PM^{DIS}. If this is a negative instance for PM^{DIS}, it is also negative for PM^D. Furthermore, if it is a positive instance for PM^{DIS}, exactly k deletions should be applied, and so no substitution or insertion are used in that solution. Hence, that solution is also a solution to PM^D, and the reduction holds. ◀

The same result in fact holds for all other distances, and in particular the longest common sub-word distance ID. This proves once again the result shown in [26]:

▶ **Corollary 14.** *The problems PM^I, PM^{DI}, PM^{IS}, PM^{DS} are all* NP*-complete.*

Proof. From Section 3.2, PM^I and PM^D are equivalent in the 1-to-1 case. For the other problem, we do an immediate reduction from PM^I or PM^D analog to the proof of Theorem 13. ◀

We now turn to proofs of NP-completeness without the restriction asking f to be injective.

4.3 Function Matching FM_1^d

The problem considered in this section is the one where both deletions and f are applied to the first word. A reduction very similar to the one given for PM^D is used.

▶ **Theorem 15.** *FM_1^D is* NP*-complete.*

Proof idea: The reduction follows the same idea as in Theorem 12. Since the function f realizing the matchings is not injective in this version, it will be used to send every vertex to its color. Moreover, we add more "sink" $\perp$ letters to force the image of every unused letter. The full proof can be found in Appendix A.

This again ensures the NP-completeness of the problem for all edit distances, using the same proof as for Theorem 13.

▶ **Corollary 16.** *The problem FM_1^{DIS} of function matching under the Levenshtein distance is* NP*-complete. The problems FM_1^I, FM_1^{ID}, FM_1^{IS}, FM_1^{DS} are all* NP*-complete too.*

We can notice that the problem FM_1^S, where substitution is the only operation allowed, is polynomial-time solvable. Intuitively, for each parameter, consider the possible parameters that it could be mapped to, and their respective number of occurrences. Then, choose the letter with the highest number of occurrences for the mapping. The remaining letters are then substituted.

4.4 Function Matching FM_2^d

The problem considered in this section is the one where deletions are applied to the second word, while f is applied to the first word.

▶ **Theorem 17.** *FM_2^D is NP-complete.*

Proof Idea: The proof is very similar to the previous case, but the bracketing has to be adapted. Separators Y^e are duplicated enough times to ensure that no vertex variable is mapped to them. The full proof can be found in Appendix A.

▶ **Corollary 18.** *FM_1^I, FM_2^{DI}, and FM_2^L are all NP-complete.*

Proof. FM_1^I is equivalent to FM_2^D. For the two other problems, we use a reduction from FM_2^D exactly like in Corollary 14. ◀

This last result completes the picture of NP-completeness proofs, and indicates that computing the distances between parameterized words defined in Section 3.3 is in general an NP-complete problem.

Similarly to FM_1^S, FM_2^S is also polynomial-time solvable.

5 Approaches to Solve Parameterized Matching

In this section, we discuss two ways to get around the difficulty of the parameterized matching problems. The first one is to design an FPT algorithm in the alphabet size, and the second one is to translate the problem into a SAT formalism, with the intent of using a SAT-solver.

5.1 An FPT Algorithm in the Alphabet Size

The fact that Σ and Π are part of the input is what makes the various parameterized matching problems NP-hard. When the alphabet size is considered fixed, a simple polynomial algorithm can be used, which generalizes the "naïve" brute force algorithm of Theorem 1 of [26]:

Algorithm 1 Simple FPT algorithm for FM^d

```
m ← 0
for all functions f : Π → Π do
    dist ← d(f(u), v)
    if dist ≤ m then
        m ← dist
    end if
end for
```

▶ **Theorem 19.** *Let d be a distance. Algorithm 1 computes $FM^d(u,v)$ in time $O(|\Pi|^{|\Pi|}|u||v|)$*

Proof. Algorithm 1 uses an exhaustive search and finds $\min_{f:\Pi\to\Pi} d(f(u),v)$, which is the definition of $FM^d(u,v)$. Furthermore, there are $|\Pi|^{|\Pi|}$ functions from Π to Π, and computing $d(f(u),v)$ is done in time $O(|f(u)||v|) = O(|u||v|)$, hence a total running time in $O(|\Pi|^{|\Pi|}|u||v|)$. ◀

Note that this also leads to a similar algorithm for PM^d by iterating over injective functions rather than all functions from Π to Π.

5.2 A MaxSat Formulation of Parameterized Matching

In this section, we encode PM^d problems into SAT problems, with the intent of solving them with a SAT solver. More precisely, we will use the weighted max-SAT variant of SAT, which is defined in the following way :

- **Input:** a set V of literals, a formula $\varphi = \bigwedge_{i=1}^{n} \varphi_i$ on V in conjunctive normal form (CNF), a weight function $w : [\![1, n]\!] \to \mathbb{N}$.
- **Output :** a valuation $\nu : V \to \{0, 1\}$ such that $\sum_{\nu \vDash \varphi_i} w(i)$ is maximal.

Moreover, we will sometimes use a partially weighted variant of Max-SAT, which is defined in the following way :

- **Input:** a set V of literals, a satisfiable formula φ_c on V in CNF, a formula $\varphi_w = \bigwedge_{i=1}^{n} \varphi_i$ on V in CNF and a weight function $w : [\![1, n]\!] \to \mathbb{N}$.
- **Output :** a valuation $\nu : V \to \{0, 1\}$ such that $\nu \vDash \varphi_c$ and $\sum_{\nu \vDash \varphi_i} w(i)$ is maximal.

In that case, clauses of φ_c are called "hard" clauses while clauses of φ_w are called "soft clauses". We give a proof of the equivalence in Proposition 26 of Appendix C.

We will define an encoding of an instance (u, v) of PM^d such that an assignment of the variables of V will define an alignment between u and v. First, we make a link between the ID edit distance and the length of the optimal alignment between two strings.

▶ **Definition 20.** *Let u and v be two words on Π, such that $p = |u|$ and $p' = |v|$. A set $A \subset [\![1, |u|]\!] \times [\![1, |v|]\!]$ is an alignment between u and v iff the following are true:*

1. *Each position of u appears at most once : For all $1 \leq i \leq p$ and $1 \leq j, j' \leq p'$, if $(i, j) \in A$ and $(i, j') \in A$, then $j = j'$.*
2. *Each position of v appears at most once : For all $1 \leq j \leq p'$ and $1 \leq i, i' \leq p$, if $(i, j) \in A$ and $(i', j) \in A$, then $i = i'$.*
3. *There are no crossings : if $(i, j) \in A, (i', j') \in A$, and $i' > i$, then $j' > j$.*
4. *Aligned positions match in u and v : if $(i, j) \in A$, then $u_i = v_j$*

An alignment relates to the insertion/deletion distance ID in the following way :

▶ **Theorem 21.** *Let u, v be words on Π and $k \leq |u| + |v|$ be an integer. The following are equivalent :*

1. *There exists an alignment A such that $2|A| = |u| + |v| - k$*
2. *$ID(u, v) \leq k$.*

Proof. The proof, which works by induction, can be found in Appendix C. ◀

We now turn to the max-SAT encoding of our problem.

▶ **Theorem 22.** *Let u and v be two words over Π. There exists a formula $\varphi_{u,v} = \varphi_c \wedge \varphi_w$ and a weight function w, instance of the partially weighted Max-SAT problem such that the following are equivalent:*

- *ν is a solution to this partially weighted Max-SAT instance and satisfies k clauses of φ_w*
- *There exists a function $f : \Pi \to \Pi$ and an alignment between $f(u)$ and v of size k.*

The formula φ uses $|m||p| + |\Pi|^2$ variables and is of size $O(m^2p^2)$, where $m = |u|$ and $p = |v|$. Moreover, there exists φ^{inj} of size $O(|\Pi|^3)$ such that the above result is true for f injective by replacing φ_c with $\varphi'_c = \varphi_c \wedge \varphi^{inj}$.

In particular, finding the valuation maximizing k gives a maximal alignment between u and v, and with Theorem 21, the distance $ID(u, v)$.

Proof. For this proof, we fix an ordering on the alphabet $\Pi = \{a_1, \ldots, a_n\}$.
We define the set of literals V as $V = \{x_{i,j} \mid 1 \leq i \leq |u|, 1 \leq j \leq |v|\} \cup \{y_{a,b} \mid a \in \Pi, b \in \Pi\}$. Intuitively, $x_{i,j}$ represents a match between position i and j in the alignment, and $y_{a,b}$ will represent the fact that $f(a) = b$. We define the following sets of formulas, where all indices i are taken between 1 and m and all j between 1 and p, and a and b are taken in Π :

$$\forall i\ \forall j' \neq j, \qquad \varphi^{A_1}_{i,j,j'} \equiv x_{i,j} \implies \neg x_{i,j'} \qquad \text{(NoDouble i)}$$

$$\forall j\ \forall i' \neq i, \qquad \varphi^{A_2}_{i,i',j} \equiv x_{i,j} \implies \neg x_{i',j} \qquad \text{(NoDouble j)}$$

$$\forall i' > i\ \forall j' < j, \qquad \varphi^{C}_{i',i,j,j'} \equiv x_{i,j} \implies \neg x_{i',j'} \qquad \text{(NoCrossing)}$$

$$\forall a \forall b \neq b', \qquad \varphi^{f}_{a,b,b'} \equiv y_{a,b} \implies \neg y_{a,b'} \qquad \text{(Function)}$$

$$\forall a \neq a' \forall \neq b, \qquad \varphi^{inj}_{a,a',b} \equiv y_{a,b} \implies \neg y_{a',b} \qquad \text{(Injectivity)}$$

$$\forall i \forall j, \qquad \varphi^{M}_{i,j} \equiv x_{i,j} \implies y_{u_i,v_j} \qquad \text{(Match)}$$

$$\forall i, \qquad \varphi^{\exists}_{i} \equiv \bigvee_{1 \leq j \leq p} x_{i,j} \qquad \text{(ExistsMatch)}$$

We then define φ_c as the conjunction of all the formulas (NoDouble i), (NoDouble j), (NoCrossing), (Function), and (Match). Furthermore, we define φ^{inj} as the conjunction of all the (Injectivity) formulas. Lastly, we define $\varphi_w = \bigwedge_{1 \leq i \leq m} \varphi^{\exists}_i$, and set $w(C) = 1$ for every clause C of φ_w.

There are $m\binom{p}{2}$ (NoDouble i) formulas, $p\binom{m}{2}$ (NoDouble j), $\binom{m}{2}\binom{p}{2}$ (NoCrossing), $n\binom{n}{2}$ (Function) and (Injectivity) formulas, pm (Match) formulas and n (ExistsMatch) formulas.

We now prove both implications of the theorem. Suppose ν is a valuation satisfying φ_c and k clauses of φ_w. We define, for all $a, b \in Pi$, $f(a) = b$ if and only if $\nu(y_{a,b}) = \top$. Since ν satisfies all the (Function) formulas, this is a correct definition of a (partial) function. We define $A = \{(i, j) \mid \nu(x_{i,j)=\top}\}$. A is an alignment between $f(u)$ and v. Indeed : (NoDouble i) and (NoDouble j) ensures point 1. and 2. of Definition 20, (NoCrossing) ensures point 3., and Match ensures point 4. The size of A is the number of $x_{i,j}$ instantiated to $\top$, which is exactly the number of clauses of φ_c satisfied, i.e., k.
Suppose now that there exists a function $\Pi \to \Pi$ and an alignment A between $f(u)$ and v. Similarly, we define $\nu(y_{a,b}) = \top$ if and only if $f(a) = b$, and $\nu(x_{i,j}) = \top$ if and only if $(i, j) \in A$. Since A is an alignment, ν satisfies (NoDouble i),(NoDouble j), and (NoCrossing). Since f is a function, (Function) is satisfied. Finally, if $\nu(x_{i,j}) = \top$, then $(i, j) \in A$, and since A is a matching, $f(u)_i = f(u_i) = v_j$ and $\nu(y_{u_i,v_j}) = \top$.

The proof for φ^b is the same, and (Injectivity) ensures the injectivity of f. ◀

What is more, this proof can be adapted to change the ID distance to the Levenshtein distance, simply by choosing to consider all the (Match) formulas as soft clauses.

6 Experiments

The two approaches presented in Section 5 were implemented in Python to solve PM^{ID}. They are available under the GPL license at `https://github.com/AaronFive/paramatch`. The FPT algorithm of Section 5.1 is implemented in the function `parameterizedAlignment`

of file `fpt_alphabet_size.py`. The MaxSAT-reduction of Section 5.2 is implemented in the function `make_sat_instance` of file `sat_instance.py`. The MaxHS solver [18] available at `http://www.maxhs.org` is used by our script to solve the MaxSAT instances derived from the PM^{ID} instances.

Our initial motivation to introduce parameterized matching under various distances is theater play comparison. To represent the structure of a theater play, we represent each character by a letter of the alphabet, and create the parameterized word obtained by considering the succession of all consecutive speakers. To check their adequacy with real data, we use a corpus of theater plays in which each character is represented by one letter of the alphabet, and each act of the play is represented by a string corresponding to the sequence of speaking characters. A letter may be duplicated in this string if the corresponding characters has lines in the end of a scene and in the beginning of the next one. Therefore, the edit distance between two parameter words representing acts will be small if both acts have a similar structure in terms of succession of speaking characters. We selected a corpus of 10 pairs of plays where one inspired the other, and performed 47 comparisons between pairs of acts. Among those comparisons, 26 were solved by the maxSAT algorithm and all by the FPT algorithm (detailed results are presented in the supplementary material available at `https://github.com/AaronFive/paramatch/tree/main/corpus10pairs`), with a 800 second timeout. The computation times are obtained on a XMG laptop running on Windows, with a 2.60 Ghz processor and 16 Gb RAM. Only the running time of MaxHS is provided, the encoding into a MaxSAT formula usually runs in approximately 1 second. Note that all instances are solved faster by the FPT algorithm than by the MaxSAT approach. The analysis of running times depending on the product of the lengths of the input strings (see supplementary material) shows that the MaxSAT approach may be relevant for strings with more than 10 distinct characters, but where the product of the length of input strings may not exceed 2000.

7 Conclusion

In this paper, we studied the complexity of several variants of the edit distance problem between parameterized words. We proved the NP-completeness of all previously unsolved cases, including the Levenshtein distance left open in [24], and provided practical approaches to solve real instances of those problems. We also studied similar problems for various definitions of words with parameters, namely parameter words and parameterized expressions, proving some relationships with parameterized word problems.

As future work, we will study the restrictions introduced in [21, 22] for a pattern matching problem with patterns in the parameter, in order to obtain polynomial time algorithms for the edit distance between parameterized words. Moreover, we will explore the question of distance between sets of words, in particular when they are defined through generalizations of automata. These problems are variants of the notion of distance between regular languages as defined in [12]. In this context, we can notice that different notions of automata can be considered: either automata generating parameterized words, or automata using parameters to define languages over classical words, with two different semantics as defined in [11].

References

1 Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer, 1993.

2 Amihood Amir, Yonatan Aumann, Richard Cole, Moshe Lewenstein, and Ely Porat. Function matching: Algorithms, applications, and a lower bound. In *Proceedings of the 30th International Conference on Automata, Languages and Programming*, pages 929–942, 2003. doi:10.1007/3-540-45061-0_72.

3 Amihood Amir, Martin Farach, and S. Muthukrishnan. Alphabet dependence in parameterized matching. *Information Processing Letters*, 49(3):111–115, 1994. doi:10.1016/0020-0190(94)90086-8.

4 Dana Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980. doi:10.1016/0022-0000(80)90041-0.

5 Alberto Apostolico, Péter L. Erdős, and Moshe Lewenstein. Parameterized matching with mismatches. *Journal of Discrete Algorithms*, 5(1):135–140, 2007. URL: https://www.sciencedirect.com/science/article/pii/S1570866706000256, doi:https://doi.org/10.1016/j.jda.2006.03.014.

6 Brenda S. Baker. A theory of parameterized pattern matching: Algorithms and applications. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 71–80, New York, NY, USA, 1993. Association for Computing Machinery. doi:10.1145/167088.167115.

7 Brenda S. Baker. Parameterized duplication in strings: Algorithms and an application to software maintenance. *SIAM Journal on Computing*, 26:1343–1362, 1997.

8 Brenda S. Baker. Parameterized diff. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, page 854–855, USA, 1999. Society for Industrial and Applied Mathematics.

9 Pablo Barceló, Leonid Libkin, and Juan L. Reutter. Querying graph patterns. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2011)*, pages 199–210. ACM, 2011. doi:10.1145/1989284.1989307.

10 Pablo Barceló, Juan Reutter, and Leonid Libkin. Parameterized regular expressions and their languages. *Theoretical Computer Science*, 474:21–45, 2013. doi:10.4230/LIPIcs.FSTTCS.2011.351.

11 Pablo Barceló, Leonid Libkin, and Juan Reutter. Parameterized regular expressions and their languages. *Theoretical Computer Science*, 474:21–45, 2011. doi:10.1016/j.tcs.2012.12.036.

12 Michael Benedikt, Gabriele Puppis, and Cristian Riveros. The cost of traveling between languages. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 234–245, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

13 Michael Benedikt, Gabriele Puppis, and Cristian Riveros. Regular repair of specifications. In *2011 IEEE 26th Annual Symposium on Logic in Computer Science*, pages 335–344, 2011. doi:10.1109/LICS.2011.43.

14 Michael Benedikt, Gabriele Puppis, and Cristian Riveros. Bounded repairability of word languages. *Journal of Computer and System Sciences*, 79(8):1302–1321, 2013. doi:10.1016/j.jcss.2013.06.001.

15 Francine Blanchet-Sadri. *Algorithmic Combinatorics on Partial Words (Discrete Mathematics and Its Applications)*. Chapman, Hall/CRC, 2007.

16 William W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. *SIGMOD Rec.*, 27(2):201–212, 1998. doi:10.1145/276305.276323.

17 Richard Cole, Carmit Hazay, Moshe Lewenstein, and Dekel Tsur. Two-dimensional parameterized matching. *ACM Trans. Algorithms*, 11(2), oct 2014. doi:10.1145/2650220.

18 Jessica Davies. *Solving MAXSAT by Decoupling Optimization and Satisfaction*. PhD thesis, University of Toronto, Canada, 2014. URL: http://hdl.handle.net/1807/43539.

19 Arnab Ganguly, Rahul Shah, and Sharma V. Thankachan. Pbwt: Achieving succinct data structures for parameterized pattern matching and related problems. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*,

693 pages 397–407, 2017. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611974782.
694 25, arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611974782.25, doi:10.1137/
695 1.9781611974782.25.
696 **20** M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of*
697 *NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition
698 edition, 1979.
699 **21** Pawel Gawrychowski, Florin Manea, and Stefan Siemer. Matching patterns with variables
700 under hamming distance. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International*
701 *Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27,*
702 *2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 48:1–48:24. Schloss Dagstuhl - Leibniz-
703 Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.48.
704 **22** Pawel Gawrychowski, Florin Manea, and Stefan Siemer. Matching patterns with variables
705 under edit distance. In Diego Arroyuelo and Barbara Poblete, editors, *String Processing*
706 *and Information Retrieval - 29th International Symposium, SPIRE 2022, Concepción, Chile,*
707 *November 8-10, 2022, Proceedings*, volume 13617 of *Lecture Notes in Computer Science*, pages
708 275–289. Springer, 2022. doi:10.1007/978-3-031-20643-6_20.
709 **23** Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and*
710 *Computational Biology*. Cambridge University Press, 1997.
711 **24** Carmit Hazay, Moshe Lewenstein, and Dina Sokol. Approximate parameterized matching.
712 *ACM Trans. Algorithms*, 3(3):29–es, 2007. doi:10.1145/1273340.1273345.
713 **25** Wilbert Jan Heeringa. *Measuring dialect pronunciation differences using Levenshtein distance*.
714 PhD thesis, University of Groningen, 2004.
715 **26** Orgad Keller, Tsvi Kopelowitz, and Moshe Lewenstein. On the longest common parameterized
716 subsequence. *Theoretical Computer Science*, 410(51):5347–5353, 2009. doi:10.1016/j.tcs.
717 2009.09.011.
718 **27** Lillian Jane Lee. *Similarity-based approaches to natural language processing*. PhD thesis,
719 Harvard University, 1997.
720 **28** Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and
721 reversals. In *Soviet physics doklady*, volume 10(8), pages 707–710. Soviet Union, 1966.
722 **29** Moshe Lewenstein. *Parameterized Matching*, pages 635–638. Springer US, Boston, MA, 2008.
723 doi:10.1007/978-0-387-30162-4_282.
724 **30** Florin Manea and Markus L. Schmid. Matching patterns with variables. In Robert Mercas and
725 Daniel Reidenbach, editors, *Combinatorics on Words - 12th International Conference, WORDS*
726 *2019, Loughborough, UK, September 9-13, 2019, Proceedings*, volume 11682 of *Lecture Notes*
727 *in Computer Science*, pages 1–27. Springer, 2019. doi:10.1007/978-3-030-28796-2_1.
728 **31** Juan Mendivelso, Sharma V. Thankachan, and Yoan Pinzón. A brief history of parameterized
729 matching problems. *Discrete Applied Mathematics*, 274:103–115, 2020. Stringology Algorithms.
730 doi:10.1016/j.dam.2018.07.017.
731 **32** Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms.
732 *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
733 **33** Saul B Needleman and Christian D Wunsch. A general method applicable to the search
734 for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*,
735 48(3):443–453, 1970.
736 **34** Sascha Schimke, Claus Vielhauer, and Jana Dittmann. Using adapted levenshtein distance
737 for on-line signature authentication. In *Proceedings of the 17th International Conference on*
738 *Pattern Recognition (ICPR 2004)*, volume 2, pages 931–934. IEEE, 2004.
739 **35** T. Shibuya. Generalization of a suffix tree for RNA structural pattern matching. *Algorithmica*
740 *(New York)*, 39(1):1–19, 2004. doi:10.1007/s00453-003-1067-9.
741 **36** Bernd Voigt. The partition problem for finite abelian groups. *Journal of Combinatorial Theory,*
742 *Series A*, 28(3):257–271, 1980.
743 **37** Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*,
744 21(1):168–173, 1974. doi:10.1145/321796.321811.

745 **38** Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between
746 trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

A Details of the proofs

Proof of Lemma 7. We proceed by induction on k. If $k = 0$, then u and v are parameterized matching, and $f(u) = v$, thus $d(f(u), v) = 0$. Suppose the result holds until a fixed k. Suppose $PM^d(u, v) = k + 1$. There exist f, u'' and u' such that $d(u, u'') = 1$, $d(u'', u') = k$, and $f(u') = v$. Hence $PM^d(u'', v) \leq k$, and by induction hypothesis $d(f(u''), v) \leq k$. Moreover, since $d(u, u'') = 1$, we get u'' from u by applying only one operation. We prove that regardless of this operation, $d(f(u), f(u'')) = 1$, and thus $d(f(u), v) \leq d(f(u), f(u'')) + d(f(u''), v) \leq k+1$ which will conclude the proof. There are 3 cases to consider:

- If the operation is a deletion, $u = v_1xv_2$ and $u'' = v_1v_2$ for some words v_1 and v_2 and some letter x. Then $f(u) = f(v_1)f(x)f(v_2)$ and we can obtain $f(v_1)f(v_2) = f(u'')$ by deleting $f(x)$.
- If it is an insertion, $u = v_1v_2$ and $u'' = v_1xv_2$, and we can similarly go from $f(u)$ to $f(u'')$ by inserting $f(x)$.
- If it is a substitution, $u = v_1xv_2$ and $u'' = v_1yv_2$, and we can go from $f(u)$ to $f(u'')$ by replacing $f(x)$ with $f(y)$.

Hence $d(f(u), f(u'')) = 1$, which concludes the proof for PM^d.

Since this proof does not use the fact that f is 1-to-1, it also stands for FM_1^d. ◀

Proof of Lemma 8. It is obvious that $d(zu, zv) \leq d(u, v)$, so we only prove $d(u, v) \leq d(zu, zv)$. We prove that any rewriting sequence from zu to zv can be modified such that no edit operation is applied in z. This will be enough to prove the result, as the edit sequence obtained can be seen as an edit sequence between u and v. We proceed by induction on the size of z. Suppose $|z| = 1$. Then $z = a \in \Sigma \cup \Pi$. We can consider that no character is modified twice in an edit sequence (i.e. no character is inserted and then deleted, or inserted and then substituted etc.), as that is always sub-optimal. Suppose z is modified. There are 3 possible cases:

1. There is an insertion in z, hence a word w ends up being inserted before a. Since $zv = av$ starts with a, w must start with an a, hence $w = aw'$. We insert $w'a$ to the right of z instead with the same operations. If z should be deleted or substituted, we apply the same operation to the new a instead. These operations yield the same result, and do not modify z.
2. There is a deletion in z, and hence a is deleted. Since this an optimal rewriting sequence, no a is created at that position through insertion or substitution afterwards. Since av starts with an a, u must be of the form $u = sau'$, where all the characters in s are deleted, and a isn't. Deleting sa instead of as yields the same result, and doesn't modify z.
3. There is a substitution in z, hence a is modified into a character $b \neq a$, that will not be further modified. Since av starts with a, an a has to be inserted in z, which is handled in case 1.

Hence, we can consider that every edit operations is done in u, and $d(au, av) = d(u, v)$. Suppose now that the result is proven for $|z| = k$, and let $z = az'$, with $|z'| = k$. Using the base case and the case for $|z| = k$, we have $d(zu, zv) = d(azu, azv) = d(zu, zv) = d(u, v)$, which concludes the proof. ◀

Proof of Lemma 10 . Let $\#$ be a fresh parameterized letter. Let then $u = \#^{k+1}u_1\#u_2\# \dots \#u_n$ and $v = \#^{k+1}v_1\#v_2\# \dots \#v_n$, where $\#^{k+1}$ denotes $k+1$ repetitions of the character $\#$. The proof of the reverse direction is the same as in Lemma 9, so we only prove the other one.

Assume $FM_2^D(u, v) \leq k$. Let v' and f realize this parameterized match.

We prove that $f(\#) = \#$, and that no other character is sent to $\#$ by f. Indeed, v starts with $k+1$ symbols $\#$, which ensure that v' starts with the letter $\#$. Since u starts with $\#$ and $f(u) = v'$, $f(\#) = \#$. Furthermore, this implies that since $|u|_\# = k+n$, $|f(u)|_\# = |v'|_\# \geq k+n$. Since v' is obtained from v by deletions, we have $|v'|_\# \leq |v|_\# = k+n$. Hence $|v'|_\# = k+n$ and all those inequalities are equalities, which is only the case when no $\#$ symbols is deleted from v, and that for all $x \neq \#$, $f(x) \neq \#$.

Since all the $\#$ symbols are left untouched, the rest of the proof is the same as in Lemma 9, and all of the factors u_i and v_i are parameterized matching. ◀

Proof of Lemma 11. Let $\#$ be a fresh parameterized letter, and $N = k+2$.

Let then $u = \#^N u_1 \#^N u_2 \dots \#^N u_n \#^N$ and $v = \#^N v_1 \#^N v_2 \dots \#^N v_n \#^N$. Once again, we only prove the non-trivial implication.

Suppose $FM_1^D(u,v) \leq k$, and let f and u' realize this matching. Since u starts with $k+1$ copies of $\#$, u' starts with $\#$. Since v starts with $\#$ too, $f(\#) = \#$.

We now prove that we can consider that for all $x \neq \#$, $f(x) \neq \#$. This will also imply that no $\#$ symbol is deleted from u. Let $S = \{a \in \Pi \mid f(a) = \#\}$ be the set of symbols (different from $\#$) sent to $\#$. Since $|u|_\# = |v|_\#$, the number of deleted $\#$ symbols from u is exactly $|u|_S$, hence $|u|_S \leq k$. Let us now consider the leftmost occurrence of an element of S in u', that we denote by a. The letter a appears in u in a factor of the form $\#^N w_1 a w_2 \#^N$. Since all $\#$ in v appear in blocks of size N, a must contribute to such a block, after deletions and application of f. We distinguish two cases:

1. The entirety of the word w_1 is deleted. In this case, at least one symbol $\#$ from the left $\#^N$ block is deleted; otherwise $f(\#^N)f(a) = \#^{N+1}$ would be a factor of v, which is impossible. Thus, choosing not to delete $\#$ and to delete a instead yields the same result.
2. w_1 is not deleted. Since no character from S appears to the left of a, $f(a)$ is the start of a $\#^N$ block. Furthermore, since $|u|_S \leq k$, it is not possible to form $\#^N$ with only a and w_2, and characters from the right $\#^N$ contribute to it. Hence, at least one $\#$ symbol from this right block is deleted. Like before, the same result can be obtained by not deleting it, and deleting a instead.

Either way, we can repeat this process to eliminate all occurrences of characters of S and of deletions of $\#$, which proves that we can consider that for all $x \neq \#$, $f(x) \neq \#$. Once again, we are taken back to the conditions of Lemma 9, and the rest of the proof follows. ◀

Proof of Theorem 15. We define Π like in Theorem 12, and we add the letters $\bot_1, \bot_2, \bot_3, \bot_4$ and $\bot_5$. Similarly, we define $u_1^i, v_1^i, u_\bot^i, v_\bot^i, u_2^e$, and v_2^e just like in Theorem 12. Additionally, we define for every edge e,

$$u_\bot^e = \square_1^e \square_2^e \square_3^e \square_4^e \square_5^e \square_6^e \text{ and } v_\bot^e = \bot_1 \bot_2 \bot_3 \bot_4 \bot_5.$$

We then apply Lemma 11 with

$$u_1^1, \dots u_1^n, u_\bot^1, \dots u_\bot^n, u_2^{e_1} \dots u_2^{e_m}, u_\bot^{e_1}, \dots u_\bot^{e_m}$$

and

$$v_1^1, \dots v_1^n, u_\bot^1, \dots v_\bot^n, v_2^{e_1} \dots v_2^{e_m}, v_\bot^{e_1}, \dots v_\bot^{e_m}$$

to obtain u, v, and k. We show that G is 3-colorable $\Leftrightarrow$ $FM_1^D(u,v) \leq k$.

$\Rightarrow$ Suppose G is 3 colorable. Define f like in Theorem 12 on the c_y^i and $\square_y^e$. Let e be an edge and $k_e \in [1,6]$ be the integer such that $f(\square_{k_e}^e)$ is defined. We map every remaining $\square_y^e$

in the following way:

$$f(\square_i^e) = \begin{cases} \bot_i & \text{if } i < k_e, \\ Y^e & \text{if } i = k_e, \\ \bot_{i-1} & \text{if } i > k_e. \end{cases} \quad (1)$$

It is then easy to check that $d(f(u), v) = k$, and thus $FM_1^D(u, v) \leq k$.

$\Leftarrow$ Suppose $FM_1^D(u, v) \leq k$, and let f and u' realize it. We define a coloring of G based on f. We note, for $1 \leq i \leq n$ and $1 \leq t \leq 3$, $col(c_t^i) = c_t$. If x_i is a vertex of G, define $c(x_i)$ to be $col(c_k^i)$, where c_k^i is the only element such that $f(c_k^i) = x_i$. We show in what follows that (1) this function definition is correct and (2) it is a valid coloring, i.e. if $e = \{x_i, x_j\}$ is an edge, $c(x_i) \neq c(x_j)$.

(1): The same points 1. and 2. from the proof of Theorem 12 apply, hence for every $1 \leq i \leq n$, exactly one element from $\{c_1^i, c_2^i, c_3^i\}$ is sent to x_i, while the two others are sent to $\bot_1^i$ and $\bot_2^i$, hence the result.

(2): Let e be an edge. The words $u_\bot^e$ and $v_\bot^e$ are in matching, which is done with exactly one deletion. Hence, there exists k_e such that

$$f(\square_i^e) = \begin{cases} \bot_i & \text{if } i < k_e, \\ \bot_{i-1} & \text{if } i > k_e. \end{cases} \quad (2)$$

Moreover, u_2^e and v_2^e are in matching. Since Y^e appears in v_2^e and all the characters in u_2^e apart from $\square_{k_e}^e$ have an image different from Y^e, $f(\square_{k_e}^e) = Y^e$. Hence, the only characters that are not suppressed from u_2^e are the two characters between the $\square_{k_e}^e$. Denoting them by c and c', the construction of the word ensures that $col(c) \neq col(c')$. Hence, if $e = \{x_i, x_j\}$, we have proven $c(x_i) \neq c(x_j)$, which is (2).

The coloring c is therefore valid, which concludes the proof. ◀

Proof of Theorem 17. Let $G = (V, E)$, with $V = \{x_1, \ldots, x_n\}$ and $\{e_1, \ldots, e_m\}$. Like in the 1-to-1 case, we construct factors u_i and v_i to encode vertex coloring. The parameter alphabet contains:

- $x_1, \ldots x_n$, corresponding to V,
- the colors c_1, c_2, c_3,
- for every $e \in E$, the delimiters Y^e,
- for every $e \in E$ and every $1 \leq i, j \leq 3$, $i \neq j$, the delimiters $Y_{i,j}^e$.

We define for $1 \leq i \leq n$, $u_1^i = x_i$ and $v_1^i = c_1c_2c_3$. If e is an edge and c_i and c_j are two colors, we denote $w^e(c_i, c_j) = Y_{i,j}^e Y_{i,j}^e Y_{i,j}^e \; c_i c_j \; Y_{i,j}^e Y_{i,j}^e Y_{i,j}^e$ For every edge $e = \{x_i, x_j\}$, we now define $u_2^e = Y^e Y^e Y^e \; x_i x_j \; Y^e Y^e Y^e$ and $v_2^e = w^e(c_1, c_2)w^e(c_1, c_3)w^e(c_2, c_1)w^e(c_2, c_3)w^e(c_3, c_1)w^e(c_3, c_2)$.

We now apply Lemma 10 with $u_1^1, \ldots u_1^n, u_2^{e_1} \ldots u_2^{e_m}$, $v_1^1, \ldots v_1^n, v_2^{e_1} \ldots v_2^{e_m}$, to obtain u and v.

$\Rightarrow$ Suppose G is 3-colorable, and let $c : V \to \{c_1, c_2, c_3\}$ be a valid coloring. Define $f|_V = c$. For every edge $e = \{x_i, x_j\}$, let s and t be such that $c(x_i) = c_s$ and $c(x_j) = c_t$. We then define $f(Y^e) = Y_{s,t}^e$. It is easy to check now that $d(f(u), v) = k$, and hence $FM_2^D(u, v) \leq k$.

$\Leftarrow$ Suppose now that $FM_2^D(u, v) \leq k$. We will show that $f|_V$ defines a 3-coloring of G, by showing that (1) for all $x \in V$, $f(x) \in \{c_1, c_2, c_3\}$ and (2) If $\{x, y\} \in E$, then $f(x) \neq f(y)$.

- Lemma 10 ensures that the words u_i and v_i are in matching, which proves (1).
- Lemma 10 also ensures that the words u^e and v^e are in matching. Let $e \in E$, with $e = x_s, x_t$. We have $|u_2^e|_{Y^e} = 6$, hence $|f(u_2^e)|_{f(Y^e)} \geq 6$. Since c_1, c_2 and c_3 each occur

exactly 4 times in v_2^e, they cannot occur 6 times after deletions, and $f(Y_e) \notin \{c_1, c_2, c_3\}$. Hence, there exist $i \neq j$ with $1 \leq i, j \leq 3$ such that $f(Y^e) = Y^e_{i,j}$. This implies that all but one of the w^e factors from v_2^e are suppressed, and that the remaining one is $w^e(c_i, c_j)$. Hence $f(x_s) = c_i$ and $f(x_t) = c_j$, which proves (2).

◀

B Encoding Constant Alphabet Σ in Π

We show why it is always possible to consider that $\Sigma = \emptyset$ for certain problems. These results use the lemmas proved in Section 4.1.

▶ **Lemma 23.** *Let d be a distance, k an integer and u and v be two parameterized words over the alphabet of constants Σ and the alphabet of parameters Π. There exist words $\tilde{u}$ and $\tilde{v}$ over the alphabet of constants $\emptyset$ and the alphabet of parameters $\Pi' = \Pi \uplus \Sigma$ such that the following are equivalent:*

- *$PM^d(u, v, k)$ is realized by f;*
- *$PM^d(\tilde{u}, \tilde{v}, k)$ is realized by f.*

In particular, this implies that if $PM^d(\tilde{u}, \tilde{v}) \leq k$, all functions f realizing this matching verify that for all $x \in \Sigma$, $f(x) = x$, and for all $x \in \Pi, f(x) \in \Pi$.

Proof. Let $N = k + 1$. If $\Sigma = \{a_1, \ldots, a_n\}$, we define z to be $a_1^N a_2^N \ldots a_n^N u$ and $\tilde{u} = zu$, $\tilde{v} = zv$. It is clear that if $PM^d(u, v) \leq k$ then $PM^d(\tilde{u}, \tilde{v}) \leq k$, by following the same operations, and applying the same renaming function.

Suppose now that $PM^d(\tilde{u}, \tilde{v}) \leq k$, and let f and u' realize it. Let $i \in [1, n]$. All the letters of u between position Ni and $N(i+1)$ are a_i. At most k of these positions can be modified with an edit operation. Since $N > k$, at least one of these positions is not modified, and thus there exists $j \in [Ni, N(i+1)]$ such that $u'_j = a_i$. Since all letters in v between position Ni and $N(i+1)$ are a_i, in particular $v_j = a_i$, and hence $f(a_i) = a_i$. This proves that for all $x \in \Sigma$, $f(x) = x$, and thus $f(z) = z$. Since f is 1-to-1, this entails $f(\Pi) \subseteq \Pi$. By Lemma 7, $d(f(\tilde{u}), \tilde{v}) \leq k$. Hence $d(f(zu), zv) = d(zf(u), zv) \leq k$ and by Lemma 8, $d(f(u), v) \leq k$. Hence $PM^d(u, v) \leq k$. ◀

▶ Remark 24. Note that the words $\tilde{u}$ and $\tilde{v}$ have a size increased by $N\Sigma$. If less operations are considered, it is possible to reduce this overhead. For example, in the case of PM^D, we can take z to be of the form $a_1 \ldots a_n z^N$, to reduce the overhead to $N + \Sigma$.

Similarly, constants can be encoded in Π in some FM problems. We prove this result for FM_2^D, with the help of the block decomposition allowed by Lemma 10.

▶ **Lemma 25.** *Let u and v be two parameterized words over the alphabet of constants Σ and the alphabet of parameters Π. There exist words $\tilde{u}$ and $\tilde{v}$ over the alphabet of constants $\emptyset$ and the alphabet of parameters $\Pi' = \Pi \uplus \Sigma$ such that the following are equivalent:*

- *$FM_2^D(u, v, |v| - |u|)$ is realized by f;*
- *$FM_2^D(\tilde{u}, \tilde{v}, |\tilde{v}| - |\tilde{u}|)$ is realized by f.*

Proof. We write $\Sigma = \{a_1, \ldots a_n\}$ and $\Pi = \{b_1, \ldots, b_m\}$. We define $z_\Sigma = a_1 \ldots a_n$, and $z_\Pi = b_1 \ldots b_m$. Let $\tilde{u}$ and $\tilde{v}$ be the words obtained by applying Lemma 10 to $z_\Sigma, b_1, b_2, \ldots, b_m, u$ and $z_\Sigma, z_\Pi, z_\Pi, \ldots, z_\Pi, v$. If $FM_2^D(u, v, k)$ is realized by a function f, it realizes $FM_2^D(\tilde{u}, \tilde{v}, |\tilde{v}| - |\tilde{u}|)$ too. Indeed, it is enough to apply the same operations in v, and to delete all the characters but $f(b_i)$ in the i-th copy of z_Π.

Suppose now that $FM_2^D(\tilde{u}, \tilde{v}) \leq k$, and let f realize it. Then, by Lemma 10, we have:

- $D(z, f(z)) = 0$, and hence $f(z) = z$, which implies that for all $x \in \Sigma$, $f(x) = x$.
- For every $1 \leq i \leq m$, $D(z_\Pi, f(b_i)) = |\Pi| - 1$. Hence $f(b_i)$ is a character of z_Π, which is some character $b_j \in \Pi$.
- $D(v, f(u)) \leq k$.

Hence f verifies $D(f(v), u) \leq k$ and respects the conditions on Π and Σ, which implies that is also realizes $FM_2^D(u, v, k)$. ◀

The overhead to pay for this transformation is $O(|\Sigma| + |\Pi|^2 + k)$, where the term in k comes from the proof of Lemma 10.

Transposing the technique used for Lemma 25 is not sufficient to get a similar result for FM_1^D. The question thus remains open in this context.

C Proofs Regarding the Max-SAT Encoding

Proof of theorem 21. We proceed by induction on $|u| + |v|$. If $|u| + |v| = 0$, both u and v are the empty string, and the equivalence is trivial. Fix $n \in \mathcal{N}$ and suppose now that the result holds up for all words u, v such that $|u| + |v| \leq n - 1$. Let u and v be two words such that $|u| + |v| \leq n$. Without loss of generality, consider $|u| \geq |v|$.

Suppose $ID(u, v) \leq k$. Let ρ be a rewriting sequence between u and v of length k. If there is no deletion in u in ρ, there are only insertions in v, and v is a sub-word of u, and there exists another rewriting sequence ρ' only deleting letters from u. Hence, we can consider that there is at least a deletion in u in ρ. Let p be a position at which such a deletion occur, and let $a = u_p$. The word u can be written as $u = u'au''$ for some words u' and u''. Define $w = u'u''$. It holds that $d(w, v) \leq k - 1$ and $|w| = |u| - 1$. By induction, there exists an alignment A between w and v such that $2|A| = |w| + |v| - (k - 1) = |u| + |v| - k$. We define $r(i) = \begin{cases} i \text{ if } i < p \\ i - 1 \text{ if } i > p \end{cases}$, and $B = \{(r(i), j) \mid (i, j) \in A\}$. Since A is an alignment, so is B: it satisfies conditions 1 to 3 of Definition 20, and since $w_r(i) = u_i$, it also satisfies condition 4. Finally, $|B| = |A|$, hence $2|B| = |u| + |v| - k$, hence the result.

Suppose now that there exists an alignment A such that $2|A| = |u| + |v| - k$. Similarly, consider p, a position in u such that there does not exist a j with $(p, j) \in A$. If no such position exist, since $|u| \geq |v|$, $u = v$ and the result is proven. Consider w the word obtained by deleting u_p from u. It then holds that $|w| = |u| - 1$ and that $2|A| = |u| + |v| - k = |w| + |v| - (k - 1)$. Defining B in the same way as above yields an alignment between w and v of the same size, and thus by induction, $d(w, v) \leq k - 1$, and since $d(u, w) = 1$, $d(u, v) \leq k$. ◀

▶ **Proposition 26.** *Weighted Max-SAT and partial weighted Max-SAT are equivalent.*

Proof. Encoding a weighted Max-SAT instance as a partially weighted Max-SAT instance is straightforward, as we just have to choose φ_c to be empty.

Conversely, given a satisfiable CNF formula φ_c, a CNF formula φ_w, and a weight function w on the clauses of φ_w, we can define a weighted Max-Sat instance in the following way :

- We define $\varphi = \varphi_c \wedge \varphi_w$
- We set $W = 1 + \sum_{C_i \text{clause of} \varphi_c} w(C_i)$, and extend w to clauses of φ_c such that $w(C_j) = W$ for all clauses C_j of φ_c

If ν is a valuation, we denote by $w(\nu)$ the sum of the weights of all clauses it satisfies $\sum_{\nu \vDash C_i} w(C_i)$. Since φ_c is satisfiable, there exists a valuation ν_c such that $\nu_c \vDash \varphi_c$, and $w(\nu_c) \geq |\varphi_c|W$. Let now ν be a valuation no satisfying a clause of φ_c. Then $w(\nu_c) \leq (|\varphi_c| - 1)W + (W - 1) < w(\nu_c)$, hence nu_c is not maximal and cannot be a solution to the weighted Max-SAT instance. ◀