Algorithmes bio-inspirés pour la traduction automatique statistique

Ameur Douib

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Algorithmes bio-inspirés pour la traduction automatique statistique

THÈSE

présentée et soutenue publiquement le 01 février 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Ameur Douib

Composition du jury

Rapporteurs : Pr Yannick Estève : Université Avignon, France
Pr Yves Lepage : Université de Waseda, Japon

Examinateurs : Pr Ammar Oulamara : Université de Lorraine, France
Pr Violaine Prince : Université Montpellier 2, France
Pr Kamel Smaïli : Université de Lorraine, France
MCF David Langlois : Université de Lorraine, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

J'aimerais tout d'abord remercier mon directeur de thèse, Kamel Smaïli, et mon codirecteur, David Langlois, pour m'avoir donné la possibilité de travailler sur ce sujet de recherche et de m'avoir accompagné durant ces 4 années afin d'arriver à ces résultats.

Je remercie également, mes amis et collègues, qui ont toujours été à mes côtés quand j'en avais besoin.

Je remercie ma famille, mes parents, mes soeurs et mon frère et en particulier je remercie ma tante Nora, mes soutiens de toujours depuis mon enfance et mes débuts en Algérie jusqu'au jour où j'ai mis les pieds en France.

Mes remerciements les plus chaleureux et les plus profonds vont à ma femme chérie, qui a eu la force de m'accepter, de m'accompagner, de m'encourager et de me supporter durant toutes les épreuves.

"Dieu merci"

Je dédie cette thèse à "$Tahani$" chérie

Sommaire

Bibliographie

Table des figures

Liste des tableaux

Introduction générale

Nous vivons dans un monde multiculturel, mondialisé et sans frontières de savoir, dans un monde où les connaissances, en particulier textuelles (articles, livres, sites web, etc.), sont largement disponibles et accessibles en grande partie sur internet. Ces connaissances, estimées à des milliards de téra-octets, découlent de diverses sources et dans différentes langues. Dans ce monde, l'homme est assoiffé de ce genre de connaissances et demande de plus en plus à les acquérir de manière instantanée et ce, en cherchant à s'affranchir de la barrière de la langue. C'est en effet le cas pour la recherche de sous-titres, dans sa propre langue, pour des séries étrangères, les discussions (courrier électronique ou messagerie instantanée) entre interlocuteurs parlant différentes langues, ou encore, l'accès aux contenus de sites web écrits dans d'autres langues que la sienne.

Afin de pouvoir satisfaire ces besoins et mettre à disposition des gens toute cette masse de données, dans les différentes langues utilisées à travers le monde, un solide mécanisme de traduction automatique doit être mis en place. Un tel mécanisme de traduction permet également de faciliter la communication et les échanges entre nations et cultures, dans leurs différents aspects, professionnel, économique, culturel et éducatif.

La traduction automatique (TA) par ordinateur, est l'un des plus importants axes de recherche en traitement automatique du langage naturel (TALN), un domaine pour lequel un grand nombre de travaux de recherche sont publiés chaque année [Bojar et al., 2017] pour proposer de nouvelles approches, de nouveaux systèmes et pour améliorer les systèmes existants, afin d'améliorer les performances de traduction dans les différentes langues qui existent dans notre monde. L'automatisation du processus de traduction, qui est à la base une tâche humaine, consiste à modéliser tout un ensemble de connaissances linguistiques humaines pour les réutiliser, par la suite, afin de traduire un texte dans une langue source vers une langue cible, de manière automatique par ordinateur, sans avoir recours à une expertise humaine. Les approches et systèmes proposés, depuis le début du siècle précédent, se différencient selon les connaissances humaines que l'on veut modéliser et selon les algorithmes utilisés pour la mise en place des systèmes [Vauquois and Boitet, 1985].

La traduction automatique, et ses différentes approches, ont évolué dans le temps, en parallèle avec l'évolution et le développement des ressources et des technologies, pour converger vers des approches basées sur les corpus, connues sous le nom d'*"Approche Empirique"*. Cette approche consiste à apprendre de manière empirique un grand nombre d'évènements et de relations linguistiques entre deux langues à partir des corpus de textes parallèles, dans lesquels chaque phrase dans le corpus source est alignée avec une phrase dans le corpus cible. La grande disponibilité de ces données textuelles parallèles, dans différentes langues et à travers plusieurs organismes, comme l'Union Européenne ou les Nation Unies, a permis aux approches empiriques de se développer rapidement et de proposer des systèmes présentant des performances de traduction très

intéressantes.

Parmi les différentes approches empiriques qui existent dans la littérature, la traduction automatique statistique (TAS) [Al-Onaizan et al., 1999], [Och, 2002], est l'approche la plus utilisée ; elle propose les meilleures performances de traduction, en particulier la traduction automatique statistique à base de segments [Koehn et al., 2003]. Dans un tel système, un ensemble de modèles, essentiellement les modèles de traduction (MT) et de langage (ML), sont formés après application de différentes analyses statistiques sur des corpus de textes parallèles. Différentes informations et relations de traduction, détectées entre la langue source et la langue cible, sont sauvegardées dans ces modèles, afin de les réutiliser par la suite pour la construction et l'évaluation des traductions dans le décodeur du système.

Au niveau du décodeur d'un système de TAS, le problème de construction des traductions (décodage) et le problème d'optimisation des poids de la fonction log-linéaire pondérée [Och and Ney, 2003], utilisée pour combiner plusieurs scores afin d'évaluer les hypothèses de traduction au cours du décodage, sont deux problèmes d'optimisation qui doivent être résolus afin de produire des traductions de bonne qualité.

L'algorithme de décodage de référence est basé sur un algorithme de recherche en faisceau [Koehn et al., 2007], où la traduction est construite de manière incrémentale. Cet algorithme a l'avantage de tester un grand nombre d'hypothèses de traduction, mais il est contraint d'appliquer un processus d'élagage afin d'éviter l'explosion combinatoire. Cet élagage est appliqué en évaluant la qualité d'hypothèses de traduction partielles et non complètes ; une prise de décision non optimale qui peut faire éliminer des hypothèses potentiellement bonnes une fois étendues complètement. Quant aux poids de la fonction log-linéaire, ils sont optimisés par un algorithme de recherche linéaire [Och, 2003] dans un processus itératif. Le jeu de poids optimal est celui qui permet au décodeur de produire un ensemble de traductions qui maximise un score d'estimation de la qualité des traductions (ex : BLEU [Papineni et al., 2002]) en les comparant à des traductions de référence, sur un ensemble de développement.

Pour pallier ce problème de prise de décision au cours du décodage, la manipulation d'hypothèses de traduction complètes et non partielles [Langlais et al., 2007] peut être une solution pertinente. En effet, cela permet d'avoir des scores d'évaluation de solutions complètes ce qui doit minimiser le risque d'éliminer de bonnes solutions au cours du décodage. Cette solution permet aussi de se focaliser davantage sur la fonction d'évaluation afin d'améliorer le processus d'évaluation des solutions, soit en rajoutant de nouveaux scores dans la combinaison de la fonction log-linéaire, soit en proposant de nouvelles approches d'évaluation.

Dans cette thèse, nous menons un travail de recherche afin d'étudier l'utilisation des algorithmes génétiques [Holland, 1973] pour la résolution des problèmes d'optimisation au niveau du décodeur. Dans ce contexte, nous proposons un nouveau système de traduction avec un décodeur entièrement basé sur des algorithmes génétiques. En effet, pour le décodage, nous proposons GAMaT *"Genetic Algorithm for Machine Translation"* [Douib et al., 2016], un algorithme dans lequel nous redéfinissons tous les mécanismes des algorithmes génétiques afin de les adapter au problème de décodage. Pour l'optimisation des poids, nous proposons GAWO *"Genetic Algorithm for Weights Optimization"* [Douib et al., 2017a], une implémentation classique des algorithmes génétiques manipulant des vecteurs de valeurs numériques.

Les algorithmes génétiques sont des algorithmes d'optimisation bio-inspirés connus pour leur efficacité à trouver le bon équilibre entre exploration et exploitation de l'espace de recherche, et qui sont appliqués souvent pour la résolution de problèmes d'optimisation complexes [Binitha and Sathya, 2012] ; c'est pour cela que nous avons décidé de les utiliser et de les adapter pour la résolution de ces deux problèmes d'optimisation. Dans la littérature de la TA [Hüe, 1997], les algorithmes génétiques ne sont utilisés que dans de rares travaux de recherche pour traiter certaines composantes de différents systèmes de traduction issus de différentes approches, et à notre connaissance, ils ne sont jamais utilisés pour la TAS à base de segments.

L'application des algorithmes génétiques pour le décodage nous permettra de manipuler un grand nombre d'hypothèse de traduction complètes (une population) tout au long du processus de décodage et non pas des hypothèses partielles. Avec cette proposition, nous espérons améliorer la prise de décision dans l'évaluation des solutions, et aussi, nous espérons exploiter tout le potentiel des algorithmes génétiques pour améliorer l'exploration et l'exploitation de l'espace de recherche au cours du décodage.

Pour la fonction *fitness* du décodeur génétique GAMaT, nous proposons une nouvelle approche d'évaluation, différente de la fonction log-linéaire largement utilisée dans la littérature. L'approche que nous proposons [Douib et al., 2017b] consiste à utiliser un réseau de neurones pour l'apprentissage d'une fonction qui prédit le score BLEU d'une hypothèse de traduction en fonction de ses différents scores estimés à partir des modèles appris au cours de l'apprentissage, que nous combinions avec la fonction log-linéaire précédemment.

Dans ce manuscrit de thèse, nous présentons en détail toutes nos propositions en justifiant nos choix de conception et d'adaptation des algorithmes génétiques pour les deux problèmes d'optimisation au niveau du décodeur, ainsi que ceux de la nouvelle fonction *fitness* apprise par le réseau de neurones. Nous présentons également différentes expérimentations pour analyser le comportement et les performances de notre système de traduction génétique. L'organisation de ce manuscrit se présente comme suit :

— Dans le Chapitre 1, nous présentons un bref historique de la traduction automatique et les différentes approches de traduction qui ont été proposées. Dans ce même chapitre, nous détaillons davantage les principes de base de la traduction automatique statistique tout en expliquant comment est construit un système de TAS à base de segments. Le problème de décodage et celui de l'optimisation des poids de la fonction log-linéaire y sont expliqués de manière détaillée.

— Dans le Chapitre 2, nous introduisons les algorithmes bio-inspirés pour l'optimisation en listant les différentes catégories de ces algorithmes. Une grande partie de ce chapitre est consacrée aux algorithmes génétiques, leur déroulement et leurs principales fonctions et composantes. Nous y présentons également les plus importantes applications de ces algorithmes pour la traduction automatique.

— GAMaT, l'algorithme génétique que nous proposons comme décodeur pour la TAS à base de segments, est présenté dans le Chapitre 3, tout en détaillant nos choix de conception et nos choix d'adaptation des algorithmes génétiques pour le problème de décodage.

— Nos deux propositions pour la fonction *fitness* de GAMaT, à savoir GAWO, l'algorithme génétique pour l'optimisation des poids de la fonction log-linéaire, et la fonction de pré-

diction du score BLEU, apprise par le réseau de neurones, sont présentées en détail dans le Chapitre 4.

— Dans le Chapitre 5, nous présentons les différentes expérimentations et tests que nous avons menés pour analyser le comportement de notre système de traduction génétique GAMaT, en utilisant les deux fonctions *fitness*. Nous présentons également différents résultats comparatifs afin de situer les performances de notre système par rapport à celles des systèmes de référence.

— Dans le Chapitre 6, nous clôturons ce manuscrit par une conclusion générale dans laquelle nous analysons les performances de nos propositions et nous discutons les avantages et limites de notre travail. Nous proposons également une discussion sur les pistes d'amélioration et d'adaptation de notre travail pour de futurs travaux de recherche en traduction automatique.

1

Traduction automatique

1.1 Introduction

L'importance de la traduction dans l'évolution et le progrès de notre monde date de bien longtemps, où l'être humain a toujours œuvré pour pouvoir communiquer et échanger avec différentes civilisations et cultures. Comprendre les langues des autres et communiquer avec eux a toujours été un moyen de progression et d'acquisition de savoirs et de connaissances et un important atout de sécurité et de défense.

De manière naturelle ou humaine, un processus de traduction consiste à prendre un texte dans une langue donnée, que nous nommons *"langue source"*, et de le traduire vers une autre langue, que nous nommons *"langue cible"*. Le texte généré dans la langue cible (traduction) doit être équivalent au texte source et il doit reproduire son sens dans la langue cible. Ce traitement de traduction nécessite des connaissances linguistiques, dans les deux langues source et cible, afin de pouvoir reproduire fidèlement le sens du texte source dans la langue cible. En effet, deux langues différentes n'ont pas le même vocabulaire ni les mêmes constructions grammaticales. Dans la Tableau 1.1, nous présentons deux exemples de traduction du français vers l'anglais.

Source	Cible
Le nom et son adjectif sont inversés en anglais.	The name and its adjective are reversed in english.
La maison blanche originale.	The original white house.

TABLE 1.1 – Exemples de traductions du français vers l'anglais.

D'un point de vue humain, nous pouvons remarquer que pour traduire la première phrase en français, nous n'avons besoin que d'un dictionnaire pour traduire chaque mot du français vers l'anglais et garder le même ordre des mots, puisque la construction grammaticale est la même pour les deux phrases dans ce cas. Le deuxième exemple, contrairement au premier, nécessite plus de connaissances pour produire une traduction correcte. En anglais, les adjectifs se placent généralement, avant le nom et non pas après comme pour le français. Ces deux langues (français-anglais) sont relativement proches car il existe plusieurs similarités linguistiques dues au rapprochement culturel et géographique des origines de ces deux langues, ce qui facilite le traitement de traduction entre elles. La traduction entre d'autres paires de langues, ayant des origines totalement différentes et séparées historiquement et géographiquement, est plus difficile et nécessite des connaissances plus poussées pour pouvoir traduire le sens d'un texte source

correctement vers la cible. À titre d'exemple, nous pouvons citer la traduction entre français et japonais, deux langues qui ont des structures et règles de construction totalement différentes.

Pour réaliser ce traitement de traduction, l'expert humain utilise ses propres connaissances linguistiques et son expérience personnelle, ainsi que d'autres ressources à l'image des dictionnaires. Cependant, le recours à l'expertise humaine, pour la réalisation de ce traitement de traduction, dans le cadre du monde actuel, s'avère coûteux en temps et en argent et ne peut répondre à toutes les attentes et exigences. L'automatisation de ce processus de traduction par ordinateur est une solution intéressante à ce problème, mais qui reste une tâche très difficile et complexe. Automatiser le processus de traduction inclut l'automatisation de toutes les connaissances et le raisonnement de l'expert humain, ainsi que la notion d'expérience, et cela de manière directe ou indirecte.

L'automatisation du processus de traduction, communément nommée traduction automatique (TA) par ordinateur, est un important axe de recherche en traitement automatique des langues naturelles (TALN) par ordinateur, qui s'est largement développé depuis le début du siècle précédent. Un grand nombre de travaux, dédiés à ce domaine, sont disponibles dans la littérature, et proposent différentes approches et techniques afin d'automatiser le traitement de traduction.

Dans ce premier chapitre, nous présentons d'une manière globale l'état de l'art des principaux travaux menés dans le domaine de la traduction automatique (TA). Nous commençons par un bref historique de l'évolution des recherches en TA et dans un second temps, nous décrivons les deux grandes familles d'approches de la TA, à savoir les approches classiques dites expertes, et les approches modernes dites empiriques largement utilisées de nos jours. Par la suite, et comme le système de traduction que nous proposons dans ce travail de thèse est basé sur un apprentissage statistique, nous abordons plus en détail l'approche statistique ainsi que les différents systèmes et techniques proposés dans le cadre de cette approche. À la fin de ce chapitre, nous présentons les principales techniques d'évaluation des traductions. Cette évaluation est d'une grande importance dans l'évaluation des systèmes de traduction et les travaux en TA.

1.2 Historique

Durant le XX^{e} siècle, les conflits géopolitiques entre les puissances mondiales et la forte concurrence militaire entre elles ont débouché sur une évolution et un développement remarquables dans plusieurs domaines en science. Pour ces mêmes raisons militaires, les scientifiques de ces pays ont profité du développement des outils informatiques pour l'automatisation du processus de traduction afin de traduire et de communiquer d'une langue vers une autre sans passer par les traducteurs humains. En effet, en 1954, en pleine guerre froide, dans des travaux de l'université de Georgetown et de IBM est proposé le premier système de TA [Hutchins, 1995]. Ce système ne traduisait que quelques phrases du russe vers l'anglais en se basant sur un ensemble restreint de règles de grammaire et avec un vocabulaire ne dépassant pas les trois cents mots. Toutefois, ce système a marqué les esprits et depuis, plusieurs pays et chercheurs se sont intéressés à la TA en injectant d'importants investissements dans le développement de systèmes de traduction. Cet intérêt grandissant pour la TA l'a placée comme l'un des plus importants axes de recherche en traitement automatique du langage. Malgré cet enthousiasme, la faible qualité des traductions proposées par les systèmes qui existaient à ce moment et les limites des machines

à disposition des chercheurs, ont ralenti le développement de la TA [Hutchins, 2001].

La fin des années soixante-dix a connu un développement considérable des travaux de recherche en TA. Ces avancées étaient possibles grâce aux progrès et améliorations des outils et machines informatiques ainsi que la richesse des corpus textuels monolingues et bilingues dans différentes langues. De nos jours, un grand nombre de corpus composés de millions de lignes (phrases) sont disponibles grâce aux institutions internationales telles que les Nations Unies et l'Union Européenne, qui archivent leurs débats et discussions sous forme textuelle et dans différentes langues. Ces dernières années, les techniques de traitement de données et les algorithmes d'intelligence artificielle ont permis la proposition de nouvelles approches de TA à base d'apprentissage automatique sur des corpus parallèles de grande taille. Ces nouvelles approches arrivent à produire des traductions d'une qualité très intéressante et qui peuvent être utilisées sans post-édition.

1.3 Approches de traduction

De nos jours, il existe de multiples systèmes et approches de traduction automatique, cependant le principe de base reste le même pour toutes ces approches [Arnold et al., 1993]. Dans ce principe de base, une première étape d'analyse des deux langues source et cible, pour lesquelles la traduction sera réalisée, est faite afin d'apprendre et extraire des connaissances de traduction qui seront utilisées dans le processus de construction des traductions. Par la suite, pour traduire un texte de la langue source, il doit être analysé en se basant sur les connaissances de traduction précédemment extraites afin d'en extraire une certaine représentation, intermédiaire ou finale, facilitant son transfert vers la langue cible. Après génération des traductions, certaines corrections (post-édition) peuvent être appliquées sur le texte cible pour produire une traduction grammaticalement correcte dans la langue cible, et qui correspond au mieux au texte source.

La différence entre les approches et systèmes de traduction automatique réside dans les approches d'analyse des langues à traiter et du type de connaissances à apprendre et extraire. Ces approches se distinguent aussi, par le choix de représentation du texte source à traduire et les règles de transfert pour la génération de la traduction dans la langue cible.

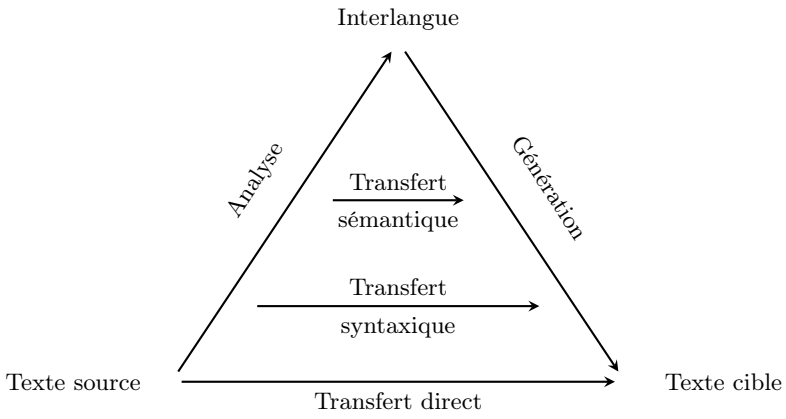


FIGURE 1.1 – triangle de vauquois, modèle pour les fondements de la traduction automatique.

Au cours des années 60, Vauquois et Boitet proposent un schéma résumant le processus de la traduction automatique dans un triangle qui représente les trois étapes d'analyse, de transfert et de génération [Vauquois and Boitet, 1985]. Ce triangle, connu sous le nom de triangle de vauquois, schématisé dans la figure 1.1, représente toutes les approches de la TA qu'on connaît à ce jour, où chacune d'elles traite d'une manière différente les trois étapes de traduction automatique.

Dans les points qui vont suivre, nous présentons en détail les deux principales familles d'approches de TA ainsi que les plus importants systèmes issus de chacune de ces deux familles et en situant chaque approche sur le triangle de vauquois.

1.3.1 Les approches expertes

Dans le cadre des approches expertes, l'ensemble du processus de traduction repose sur des connaissances de traduction d'experts humains, généralement des linguistes. Ces connaissances sont générées après l'analyse d'un corpus de texte dans la langue source et d'un corpus de texte dans la langue cible, comme nous pouvons le voir sur la figure 1.2. Cette analyse est faite séparément pour chaque langue et produit un ensemble de règles de représentation et de transfert. Dans le processus de traduction, l'analyse du texte à traduire est faite dans le but de générer une représentation sémantique, syntaxique ou morphosyntaxique de la source. Le niveau de cette représentation dépend du niveau d'analyse du texte implémenté dans l'approche utilisée. Par la suite, les modèles de connaissances de traduction et des règles de transfert, sont utilisés pour transférer la représentation de la source vers une représentation équivalente dans la langue cible. En se basant sur cette représentation cible, un texte est généré comme étant une traduction et cela en utilisant d'autres ressources linguistiques telles que les dictionnaires.

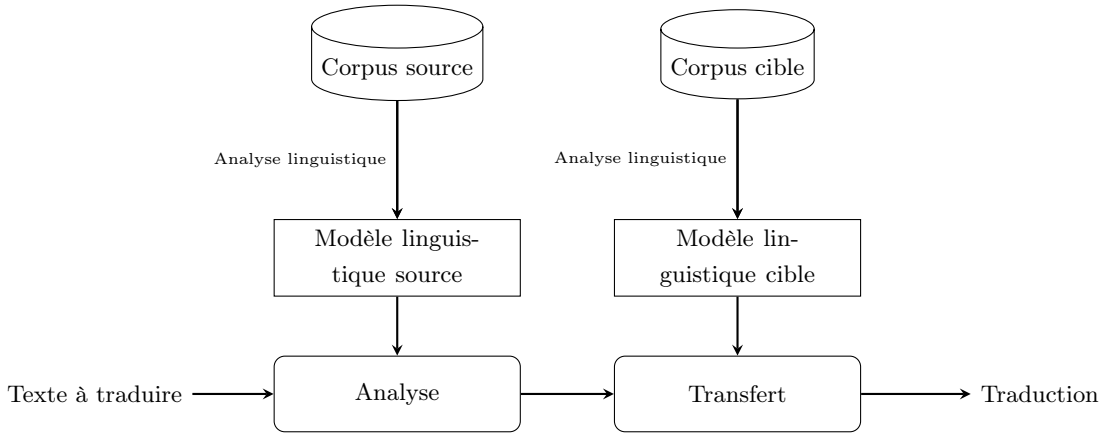


FIGURE 1.2 – Processus de traduction des approches expertes.

Dans les sections suivantes, nous présentons les principales approches expertes de la TA.

Traduction à base de dictionnaire

L'approche à base de dictionnaire est celle qui correspond au niveau le plus bas du triangle de vauquois (Figure 1.3). Dans cette approche le transfert du texte source vers la cible est réalisé directement sans appliquer d'analyse de la source. Cette approche est considérée comme la plus intuitive et la plus simple à mettre en place. Le texte source à traduire, est décomposé

en unités, généralement des mots, cette suite de mots est traduite directement en utilisant un dictionnaire bilingue. La simplicité du processus de traduction de cette approche, oblige l'intervention humaine afin de corriger les traductions produites et les rendre compréhensibles. Cette post-édition peut être une réorganisation (réordonnancement) des mots dans la langue cible, une réadaptation des traductions et même, dans certains cas, elle consiste à retraduire le texte source.

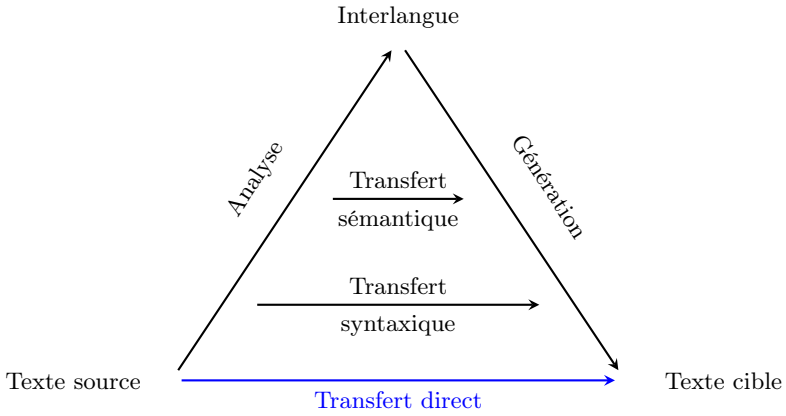


FIGURE 1.3 – triangle de vauquois, approche à base de dictionnaire

À cause de la mauvaise qualité des traductions proposées par cette approche et de la coûteuse intervention humaine, l'utilisation de la traduction à base de dictionnaire est très limitée et peu de travaux y sont consacrés, sauf dans de rares cas où le domaine de traduction est restreint avec un faible vocabulaire et une faible différence entre les deux langues de la paire à traiter.

Traduction interlangue

Comme son nom l'indique, cette approche se base sur l'utilisation d'une langue pivot pour faire le passage de la langue source vers la langue cible [Boitet, 1988]. Le processus de traduction dans le cadre de cette approche se déroule en deux étapes : la production d'une représentation du sens du texte source dans la langue pivot et la génération du texte cible à partir de cette représentation.

Comme on peut le voir dans la figure 1.4, le niveau d'analyse du texte source dans la TA à base d'interlangue est le plus élevé comparé aux autres approches. Deux modèles de connaissances de traduction sont produits dans ce cas, le premier pour le transfert de la langue source vers la langue pivot et le deuxième pour le transfert de la langue pivot vers la langue cible.

L'avantage de cette approche est le fait de rendre les modèles appris indépendants du système mis en place. Ces modèles peuvent être réutilisés pour d'autres systèmes traitant d'autres paires de langues. En effet, si un système de traduction pour la paire français-anglais est mis en place, alors la mise en place d'un système pour la paire français-arabe ne nécessite que la génération d'un modèle de transfert de la langue pivot vers l'arabe, vu que le modèle de transfert du français vers la langue pivot existe déjà.

Cette approche est souvent utilisée dans le cas où il y a un manque de ressources textuelles entre les deux langues à traiter (dialectes, langues peu dotées, etc.) et qu'il existe d'autres res-

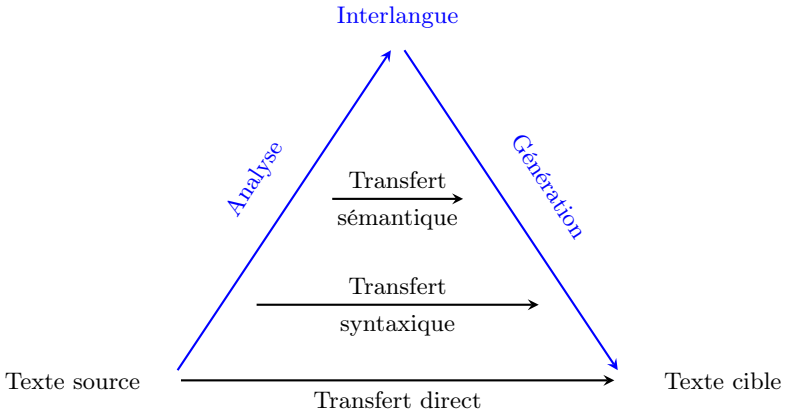


FIGURE 1.4 – triangle de vauquois, approche interlangue

sources entre la langue source et la langue pivot ainsi qu'entre la langue pivot et la langue cible. La langue anglaise, considérée comme la langue universelle et pour laquelle il existe beaucoup de ressources et travaux, est souvent utilisée comme langue pivot pour faire le passage entre deux autres langues.

L'inconvénient majeur de cette approche est que les risques de la traduction sont multipliés par deux. En effet, la représentation du texte source dans la langue pivot peut ne pas retranscrire parfaitement le sens original dans la langue source, le passage de la langue pivot vers la cible peut provoquer une autre perte supplémentaire du sens. Rajoutons à cela, le fait qu'un texte source peut avoir plusieurs sens (traductions) dans la langue pivot, donc plusieurs possibilités de traductions dans la langue cible.

Traduction à base de règles de transfert

L'approche à base de règles de transfert nécessite une analyse plus détaillée du texte source, comparée à l'analyse de l'approche directe, mais moins approfondie que celle de l'approche à base d'interlangue (Figure 1.5). Dans la TA à base de règles de transfert, et selon le niveau d'analyse implémentée, le texte source est analysé soit syntaxiquement, soit sémantiquement, soit morphosyntaxiquement ou d'une manière hybridant les trois types d'analyse.

L'étape d'analyse produit une représentation du texte source, qui est utilisée pour générer une autre représentation dans la langue cible (Figure 1.6). La génération de cette représentation cible est faite à l'aide d'une base de règles de transfert qui assurent la correspondance entre les deux représentations source et cible. Pour terminer le processus de traduction, une grammaire de la langue cible est utilisée pour générer la traduction à partir de la représentation dans la langue cible.

L'avantage de la TA à base de règles est le même que celui de l'approche précédente, à savoir la réutilisation de la grammaire et des règles de transfert pour d'autres systèmes de traduction avec d'autres paires de langues. Sauf que dans ce cas, si un système de traduction français-anglais existe, pour mettre en place un système français-arabe, il faut produire une nouvelle grammaire pour l'arabe et de nouvelles règles de transfert entre français et arabe. Cependant, comme cette approche est basée sur des analyses syntaxiques et sémantiques, il y a toujours le risque de ne

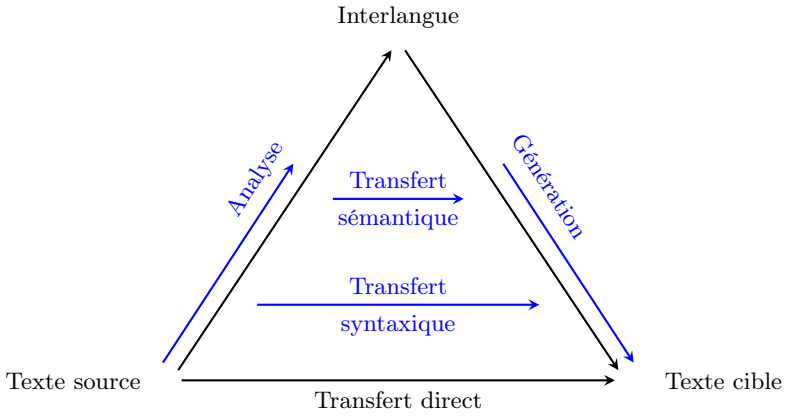


FIGURE 1.5 – triangle de vauquois, approche à base de règles de transfert.

pas pouvoir produire la bonne représentation du texte source si ce dernier est grammaticalement complexe. D'un autre côté, cette approche nécessite d'importantes connaissances linguistiques des deux langues pour pouvoir produire des grammaires et des règles de transfert permettant des traductions correctes.

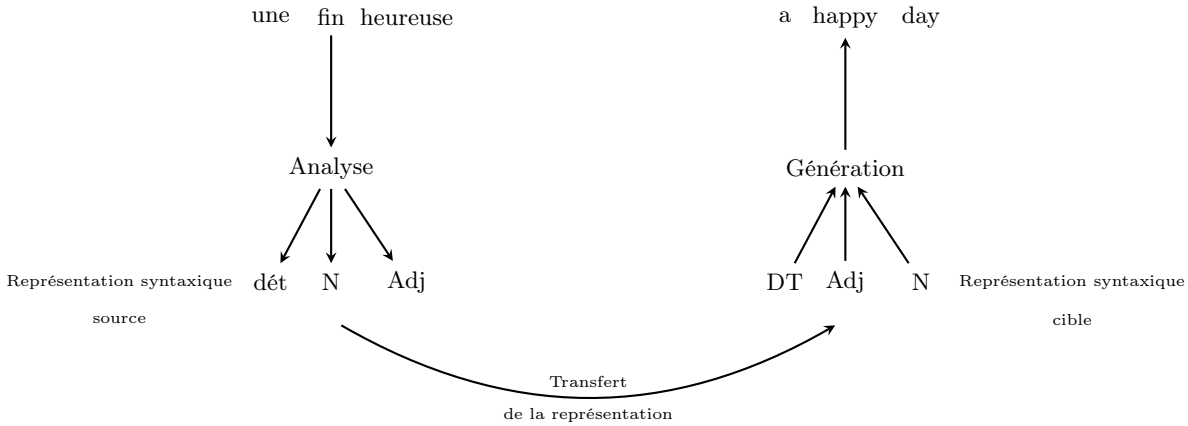


FIGURE 1.6 – Exemple de traduction avec un système à base de règles de transfert.

1.3.2 Les approches empiriques

Contrairement aux approches expertes, les approches empiriques, en traduction automatique, n'utilisent pas de connaissances de traduction d'experts humains. Les approches empiriques se basent sur la formation de modèles à travers des analyses empiriques de corpus bilingues parallèles (Figure 1.7). Ces corpus contiennent un ensemble de phrases dans la langue source et en parallèle leur traduction dans la langue cible.

L'analyse de ces corpus parallèles est faite dans le but de détecter des motifs de traduction répétitifs au niveau des mots et/ou au niveau des segments (suite de mots). Les motifs validés par le processus d'apprentissage sont, généralement, ceux qui ont une forte fréquence d'apparition dans les corpus. D'autres modèles peuvent être formés pour apprendre et détecter d'autres

connaissances de traduction qui peuvent apparaître dans les corpus d'apprentissage.

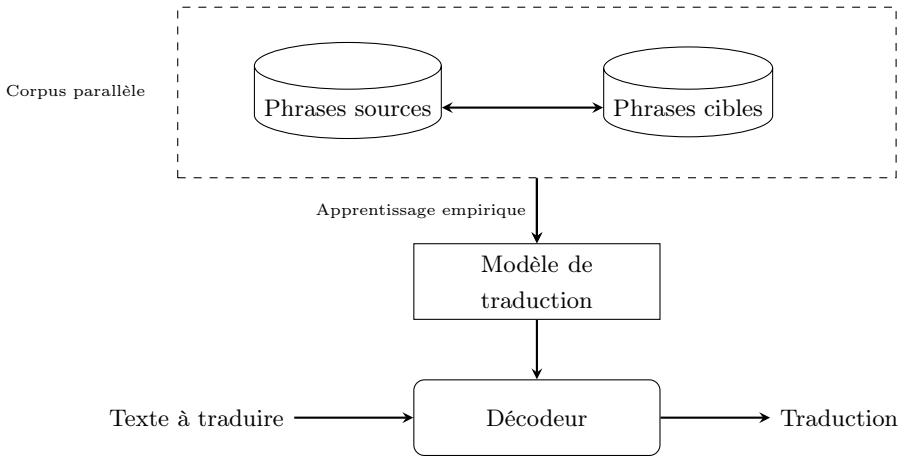


FIGURE 1.7 – Processus de traduction des approches empiriques.

Comme nous pouvons le voir sur la figure 1.7, les systèmes des approches empiriques se décomposent en deux principaux composants. Le premier, est celui de l'apprentissage, dans lequel diverses connaissances de traduction sont extraites et sauvegardées à partir des corpus. Le deuxième composant est le décodeur, dans lequel un algorithme est implémenté pour traiter le texte source afin de lui générer une traduction en sortie. Le traitement du texte source, est généralement un traitement de décomposition de la source en unités de traduction (mots ou segments). Le décodeur se base essentiellement sur les modèles formés au niveau de la phase d'apprentissage pour produire une traduction.

La principale différence entre les approches empiriques se situe au niveau de l'étape d'apprentissage des corpus parallèles, où plusieurs techniques d'apprentissage peuvent être appliquées afin d'analyser et d'extraire diverses connaissances de traduction. Dans la littérature de la traduction automatique, nous pouvons citer trois grandes approches empiriques : l'approche à base d'exemples de traductions, l'approche statistique et l'approche neuronale. Dans les points suivants, nous présentons ces trois approches en précisant les techniques d'apprentissage implémentées dans chaque approche ainsi que les connaissances extraites des corpus.

Traduction à base d'exemples de traduction

Nagao, dans son travail [Nagao, 1984], part d'une observation faite sur le processus humain de traduction. Nagao conclut que ce processus ne nécessite pas d'analyse complexe du texte source [Nagao, 1984] et se fait en deux étapes : en premier lieu, la décomposition du texte source en segments et par la suite, en se basant sur des prérequis et exemples de traduction, les segments sont traduits séparément pour en constituer à la fin, une traduction complète et correctement construite. Cette manière de traduire est connue sous le nom de traduction par analogie [Nagao, 1984]. La traduction à base d'exemples de traduction est considérée comme étant une implémentation de la traduction par analogie [Lepage, 1998].

Dans l'approche à base d'exemples, le modèle de connaissances de traduction à former, est constitué d'un ensemble d'exemples de traductions apprises après analyse du corpus de phrases

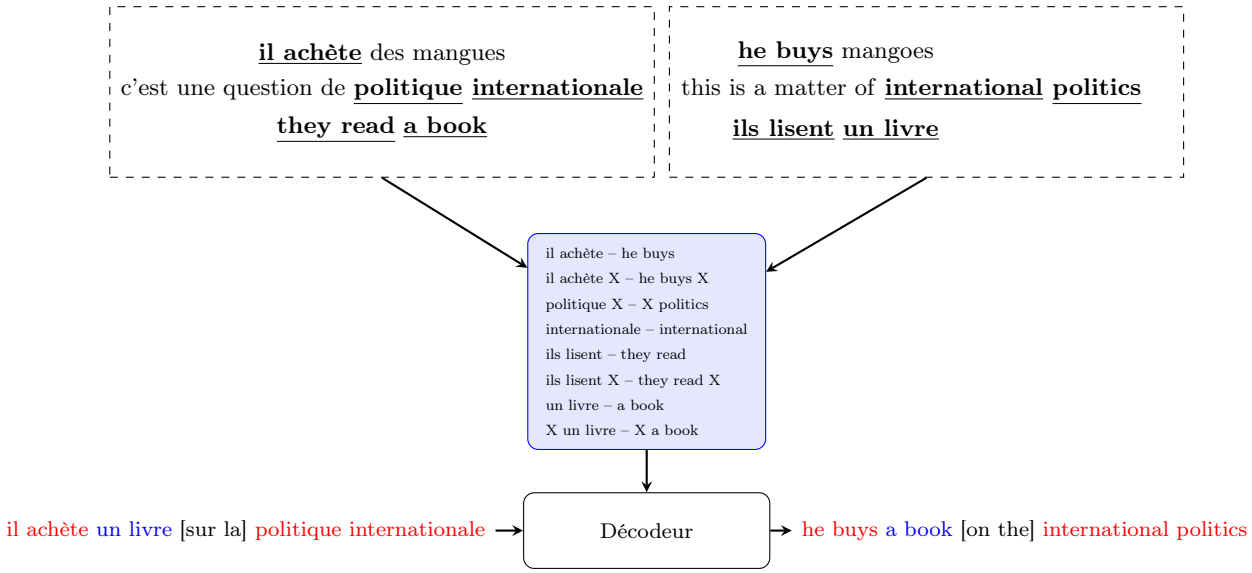


FIGURE 1.8 – Exemple de génération d'exemples de traduction et décodage.

parallèles. Cette analyse est faite à l'aide de ressources linguistiques, généralement des dictionnaires bilingues, dans le but de détecter des paires de segments (suites de mots) parallèles à partir des paires des phrases parallèles du corpus. L'apprentissage, dans cette approche, se fait ligne par ligne, sur le corpus, comme nous le présentons dans la figure 1.8, et chaque paire de segments parallèles détectée est rajoutée à la base des exemples. Plusieurs exemples de traduction peuvent être générés pour le même segment source et cela dépend de la richesse du corpus d'apprentissage.

Le processus de décodage (voir Figure 1.8) dans cette approche, se déroule en trois étapes [Nagao, 1984]. La première étape est la mise en correspondance (Matching), qui consiste à détecter dans le texte source les segments, ou idéalement la globalité du texte source, qui existent dans la base d'exemples. La deuxième étape est l'extraction des segments cibles, à partir de la base d'exemples, qui correspondent aux segments sources précédemment détectés. La troisième et dernière étape est la réorganisation des segments dans la cible afin d'éliminer les redondances de traduction, causées par les segments superposés, et afin de produire une traduction finale grammaticalement correcte.

L'inconvénient majeur de cette approche est le risque de ne pas pouvoir traduire s'il n'y a pas d'exemples de traduction dans la base qui correspondent au texte source. Pour minimiser ce risque, les systèmes à base d'exemples de traductions utilisent souvent des connaissances linguistiques et des règles de transfert. Ces systèmes sont souvent hybrides, combinant l'approche à base d'exemples et des connaissances d'approches expertes.

Approches statistiques

L'approche de traduction automatique statistique (TAS) a vu le jour vers les années 90 et durant ces deux dernières décennies une grande partie des travaux de recherche en traduction automatique lui sont consacrés.

L'approche statistique de traduction automatique se base essentiellement sur deux modèles

statistiques [Brown et al., 1991] : le modèle de traduction, qui est généré après une analyse empirique du corpus de phrases parallèles et qui est utilisé pour produire des traductions et pour évaluer le niveau de correspondance entre la source et la traduction ; le modèle de langue, qui est généré après une analyse empirique du corpus des phrases cibles uniquement, et qui est utilisé pour évaluer la pertinence d'une hypothèse de traduction dans la langue cible. En pratique, d'autres modèles sont appris et combinés avec les deux premiers [Zens et al., 2002] et cela dans le but d'améliorer le processus d'évaluation des hypothèses de traduction. L'apprentissage de ces modèles statistiques se base sur le calcul des fréquences d'apparition des motifs de traduction, en particulier les paires de mots ou de segments parallèles. Ces modèles contiennent l'ensemble des connaissances extraites à partir des corpus d'apprentissage, avec une probabilité associée à chaque connaissance indiquant sa pertinence.

Au niveau du décodeur, la traduction finale ($\hat{e}$) d'un texte source (f) doit maximiser la probabilité conditionnelle $P(e|f)$. Cette probabilité conditionnelle est estimée en se basant sur les probabilités sauvegardées dans les modèles formés auparavant. L'approche statistique est présentée plus en détail dans la section 1.4.1 en développant les deux parties d'apprentissage et de décodage.

Depuis des années, l'approche statistique propose les meilleures performances de traduction dans les travaux de recherche. Elle est à la base des principaux systèmes de traduction automatique [Callison-Burch et al., 2010, Bojar et al., 2015]. Cela a été rendu possible grâce à la facilité de mise en place de tels systèmes *via* différents outils disponibles en ligne tel que MOSES [Och and Ney, 2003, Koehn et al., 2007] et aussi grâce à la disponibilité de corpus parallèles de plus en plus grands et pour différentes paires de langues. Les approches neuronales, que nous présentons dans le point suivant, sont les approches les plus récentes qui concurrencent les approches statistiques en matière de performances de traduction.

Approches neuronales

Ces dernières années, les systèmes d'apprentissage automatique à base de réseaux de neurones simples, ou profonds, ont montré d'incroyables performances et cela dans divers domaines. Leur utilisation en traduction automatique s'est faite progressivement durant ces quinze dernières années. En effet, les premières propositions consistaient à modéliser certaines parties d'un système de TA tel que le modèle de langue [Mikolov et al., 2010] ou le modèle de traduction [Devlin et al., 2014] pour les systèmes à base d'approche statistique. Ce n'est que récemment, qu'a été proposé un système de TA conçu entièrement à base de réseaux de neurones [Sutskever et al., 2014]. Dans ce système toute la partie apprentissage sur les corpus parallèles est faite à travers un seul réseau de neurones comme celui que nous présentons dans la figure 1.9, que nous détaillons dans ce qui suit.

L'analyse du corpus parallèle dans les approches neuronales se fait à travers un réseau de neurones, généralement profond avec plusieurs couches cachées. Le modèle appris par ce réseau de neurones arrive à inclure implicitement plusieurs modèles générés séparément par les approches statistiques ou expertes.

Techniquement parlant, et comme nous l'illustrons dans la figure 1.9, la construction d'un système de traduction à base de réseaux de neurones nécessite l'utilisation des modèles de correspondance séquence-à-séquence [Sutskever et al., 2014], où dans le cas de la traduction auto-

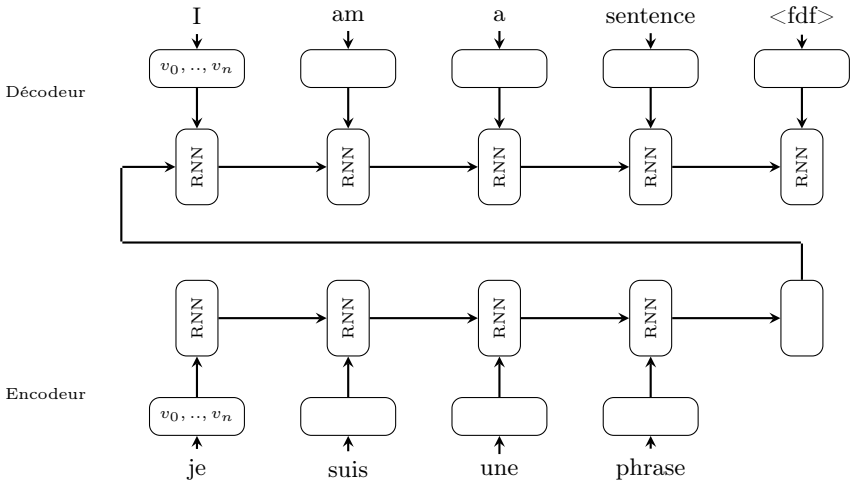


FIGURE 1.9 – Exemple de réseau de neurones récurrent pour la traduction automatique neuronale.

matique, les deux séquences à relier sont la phrase source et sa traduction parallèle, *"je suis une phrase" et "I am a sentence"* dans l'exemple de la figure 1.9. Ces modèles sont basés sur des systèmes d'encodage et de décodage du texte pour passer de la représentation symbolique (texte) à une représentation numérique, et inversement. Cette représentation numérique des données textuelles est facilement malléable par les réseaux de neurones. L'encodeur prend le texte source, sous forme de suite de mots, et génère sa représentation numérique. Le décodeur, quant à lui, prend une représentation numérique produite par le modèle et génère le texte correspondant en sortie. De nos jours, il existe différents systèmes de traduction neuronaux, dans lesquels l'encodeur, le décodeur et l'apprentissage se font *via* des réseaux de neuronaux profonds et récurrents (RNNs) [Elman, 1990], des réseaux de neurones LSTM [Hochreiter and Schmidhuber, 1997] ou encore des réseaux de neurones GRUs [Chung et al., 2014]. La différence entre ces architectures neuronales réside dans la technique de représentation numérique du texte.

Le processus de décodage dans cette approche, se rapproche de celui des approches statistiques, où la construction de la traduction se fait d'une manière incrémentale en générant une suite de mots. Au niveau du décodeur, un algorithme de recherche en faisceau est implémenté afin de produire la traduction. Dans ce processus de décodage, la traduction finale doit maximiser un score conditionnel qui est estimé grâce au modèle appris par le réseau de neurones.

Un des plus importants problèmes de cette approche est la perte de l'information du contexte des mots. En effet, ces systèmes ont du mal à représenter correctement les alignements entre les mots de la source et les mots de la cible, en particulier dans le cas des phrases longues, ce qui détériore la qualité de l'apprentissage. Pour pallier ce problème, des modèles d'attention [Bahdanau et al., 2014] sont rajoutés dans l'architecture du réseau de neurones afin de modéliser les alignements qui peuvent exister entre les mots sources et les mots cibles. Ces modèles d'attention arrivent à améliorer les performances de traduction des systèmes neuronaux. D'un autre côté, le processus d'apprentissage neuronal nécessite un nombre important de données pour arriver à des performances de traduction équivalentes à celles des approches statistiques [Koehn and Knowles, 2017].

1.4 Modélisation statistique de la traduction automatique

Dans cette section, nous abordons les fondements théoriques et mathématiques de base de la traduction automatique statistique ainsi que les plus importants travaux de recherche dans le cadre de cette approche. Nous présentons par la suite, le processus de décodage et différents systèmes de traduction statistique de l'état de l'art afin de situer notre système de traduction génétique.

1.4.1 Fondements de la TAS

Dans la modélisation statistique de la traduction automatique, le problème de traduction est considéré comme un problème d'apprentissage automatique. Dans ce processus d'apprentissage, des connaissances de traduction, généralement des motifs de traduction entre les deux langues source et cible, sont apprises statistiquement à partir d'un corpus bilingue de phrases parallèles. Ce sont les chercheurs de IBM qui ont proposé cette approche en 1990, pour la modélisation du processus de traduction automatique [Brown et al., 1990].

L'idée de base de ce modèle probabiliste pour la traduction [Brown et al., 1990] est que pour une phrase f, dans la langue source, n'importe quelle phrase e, dans la langue cible, est considérée comme traduction possible de f avec une probabilité conditionnelle $P(e|f)$. Cette information est interprétée comme étant la probabilité que la phrase cible e soit produite dans un processus de traduction sachant la phrase source f. De cette manière, la traduction la plus correcte de f est la phrase cible $\hat{e}$ qui maximise la probabilité $P(e|f)$. La valeur de cette probabilité conditionnelle est estimée selon les connaissances de traduction apprises à partir du corpus bilingue. Mathématiquement, la TAS est définie comme suit :

$$\hat{e} = \underset{e}{argmax}\ P(f|e) \times P(e) \tag{1.1}$$

À partir de l'équation 1.1, nous pouvons séparer les deux modèles de base dans la TAS, à savoir le modèle de traduction ($P(f|e)$) qui contient les motifs de traduction parallèles avec leur probabilité d'exactitude, et le modèle de langue ($P(e)$) qui contient des séquences de mots dans la langue cible avec leur probabilité de pertinence. Nous présentons en détail ces deux modèles dans les points qui vont suivre.

Comme nous le présentons dans la figure 1.10 la construction d'un système de traduction statistique se fait en deux étapes ; une première étape d'apprentissage statistique appliqué sur des corpus de phrases bilingues et monolingues pour la formation du modèle de traduction et du modèle de langue. La deuxième étape est celle du décodage, dans laquelle la meilleure traduction $\hat{e}$, qui maximise la probabilité $P(e|f)$, est construite sur la base des deux modèles précédemment appris.

De nos jours, dans les systèmes de TAS qui proposent les meilleures performances de traduction, d'autres modèles sont formés pour apprendre d'autres connaissances de traduction, que celles des modèles de traduction et de langage, et cela dans le but d'évaluer d'autres caractéristiques d'une hypothèse de traduction e en fonction de la phrase source f. Dans le cas où d'autres modèles sont combinés avec les modèles de traduction et de langage, la TAS est définie comme étant le produit des scores (h_i) estimés par ces modèles comme nous le présentons dans l'équation 1.2.

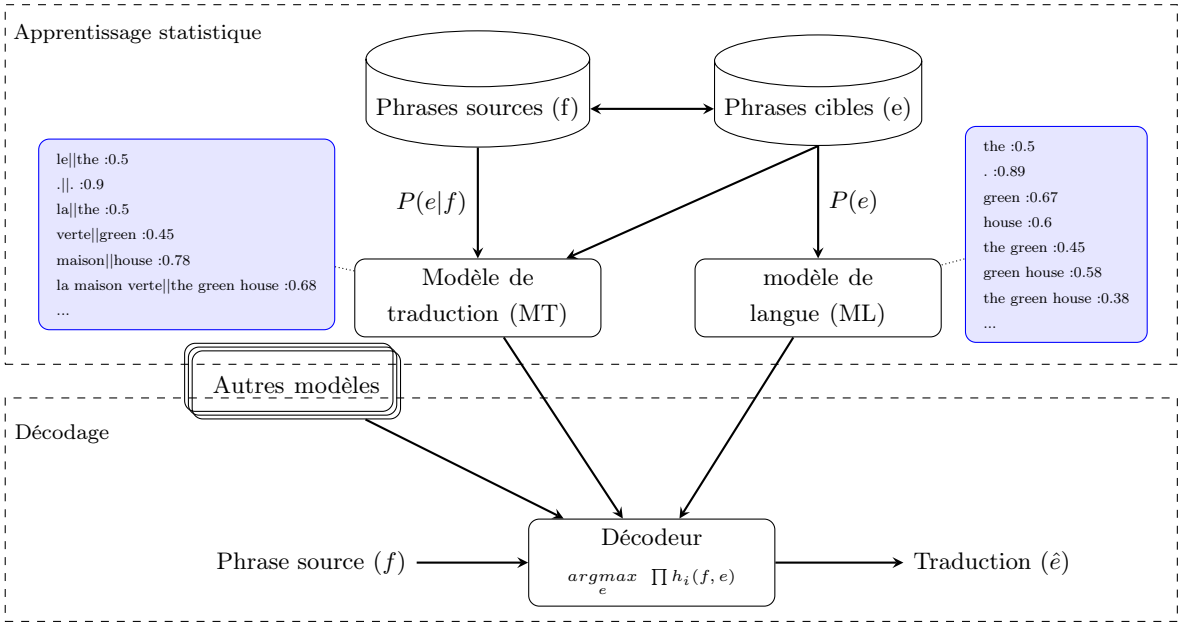


FIGURE 1.10 – Schématisation d'un système de traduction à base d'apprentissage statistique.

$$\hat{e} = \underset{e}{argmax} \prod_{i=1}^{|h|} h_i(f, e) \tag{1.2}$$

1.4.2 Modèle de traduction

C'est au niveau du modèle de traduction (MT) que sont sauvegardés les motifs de traduction résultant du processus d'apprentissage appliqué sur les deux corpus parallèles, source et cible (Figure 1.10). En effet, dans la TAS, un modèle de traduction est enregistré sous forme d'une table appelée Table de Traduction (TT). Dans cette table, des paires d'unités de traduction (mots ou segment de mots) sont sauvegardées avec une probabilité associée à chaque paire estimant la qualité de traduction entre l'unité source et l'unité cible. Dans la figure 1.10, nous pouvons voir un exemple d'une TT à base de segments apprise à partir d'un corpus parallèle français-anglais. En plus de la paire de segments et la probabilité de traduction, d'autres informations et probabilités peuvent être sauvegardées dans la TT et cela est en fonction des modèles formés et les informations que l'on veut sauvegarder.

La table de traduction peut être considérée comme un dictionnaire de traduction, dans lequel pour chaque unité source, il peut y avoir plusieurs traductions et à chaque traduction est associée une probabilité estimant sa qualité en fonction de sa fréquence d'apparition dans le corpus parallèle. À l'inverse d'un dictionnaire classique, les mots et segments dans la TT, peuvent être des noms, des noms propres, des verbes à l'infinitif ou conjugués, des abréviations, et tout autre type de mots ou segments qui existent dans les corpus d'apprentissage.

Dans la tableau 1.2, nous présentons des exemples de paires de segments extraites après apprentissage statistique appliqué sur des corpus parallèles français-anglais. Ces exemples résument les différentes paires de segments que nous pouvons avoir dans une table de traduction.

source	cible	probabilité
maison	house	0,517065
maison	houses	0,0170648
maison	home	0,196246
maison	homes	0,0784983
maison	chamber	0,00682594
maison	a house	0,00341297
maison	fox	0,001706487
ma maison	my house	0,625
ma maison	my home	0,25
ma maison	my own home	0,125
la maison blanche	the white house	0,689655
la maison blanche	as the white house is concerned	0,0344828
petit déjeuner	breakfast	0,666667
le petit déjeuner	breakfast	0,5
le petit déjeuner	my breakfast	0,5
mon petit déjeuner	my breakfast	1,0
ferai	will do	0,106095
ferai	will make	0,0677201
ferai	going to do	0,00225734
je ferai	i will do	0,123377
je ferai	i will make	0,0844156
je ferai	i am going to make	0,00324675
sommes	are	0,665836
sommes	sums	0,00894212
affaires	business	0,0283983
affaires	cases	0,0330784
affaires	things	0,01713593

TABLE 1.2 – Exemples de paires de segments, sauvegardées dans une table de traduction à base de segments.

La première remarque que nous pouvons faire est que dans une TT à base de segments, tous les types d'alignements sont possibles, à savoir : un mot source aligné à un mot cible (ex : *"maison - house"*), un mot source aligné à une suite de mots cibles (ex : *"ferai - going to do"*) et inversement (ex : *"le petit déjeuner - breakfast"*). Ce que nous pouvons remarquer aussi, c'est que la probabilité associée à chaque paire, nous informe sur la qualité de la paire en question en fonction de la fréquence d'apparition des deux segments source et cible dans des phrases parallèles du corpus d'apprentissage. En effet, si nous prenons le segment source *"maison"*, il a comme traduction la plus probable *"house"* avec une probabilité de 0.517 suivie de *"home"* avec une probabilité de 0.196. Dans la TT, d'autres traductions, inadaptées pour certaines, peuvent exister mais moins probables (ex : *"maison - fox"* - 0,0017).

Un autre évènement que nous pouvons trouver dans la TT est la traduction d'un segment source polysémique. Nous remarquons dans l'exemple de TT du Tableau 1.2, que le segment source *"sommes"* en français, peut être interprété comme étant le verbe "être" conjugué à la première personne du pluriel comme il peut être interprété comme étant une somme numérique

au pluriel. Dans la TT du Tableau 1.2, les deux traductions sont possibles mais avec des probabilités différentes et cela dépend, en premier lieu, de la qualité des corpus d'apprentissage ainsi que du domaine et du contexte des corpus. Dans cette TT, il est plus probable de traduire le mot source *"sommes"* par *"are"* que par *"sums"*. D'un autre côté, le mot source *"affaires"* a plusieurs traductions possibles ayant des probabilités qui sont proches les unes des autres. Cette faible différence entre ces probabilités de traductions résulte du fait que les fréquences d'apparition de ces paires de segments, dans le corpus d'apprentissage, sont proches les unes des autres.

La construction d'une table de traduction est basée essentiellement sur l'apprentissage des alignements entre les unités sources et cibles (paires de mots ou segments). En 1993, dans [Brown et al., 1993], on présentait les premiers modèles statistiques permettant l'apprentissage des alignements entre les mots à partir d'un corpus parallèle de phrases. Ces modèles, connus sous le nom des modèles d'IBM, sont toujours utilisés de nos jours dans la construction des TT.

Dans les premiers travaux en TAS, l'unité de base de la traduction était le mot. Dans ces travaux, la TT contenait des paires de mots sources et cibles avec la probabilité de traduction associée. Ces systèmes de traduction statistique à base de mots, ont montré leurs limites à pouvoir modéliser certains aspects de la traduction, comme l'impossibilité d'aligner un mot source à plusieurs mots cibles. Dans l'exemple de la figure 1.11, nous pouvons voir que le mot source *"partirai"* doit être aligné, d'un point de vue humain, à la suite de mots cibles (segment) *"will leave"*, cependant, cela est impossible dans le cas où le mot est l'unité de base de traduction. L'autre limite est l'impossibilité de modéliser l'alignement entre une suite de mots sources (segment) avec une suite de mots cibles. Ce dernier cas de figure est représenté dans l'exemple de la figure 1.12, où le segment source *"mode de vie extravagant"* ne peut être aligné au segment cible *"extravagant lifestyle"* alors que du point de vue humain, cet alignement est le plus correct car ce segment source français ne peut être traduit correctement en anglais en le subdivisant en mots.

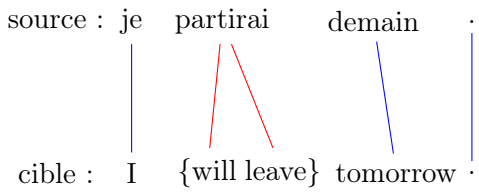


FIGURE 1.11 – Premier exemple d'alignement impossible dans un système de traduction à base de mots.

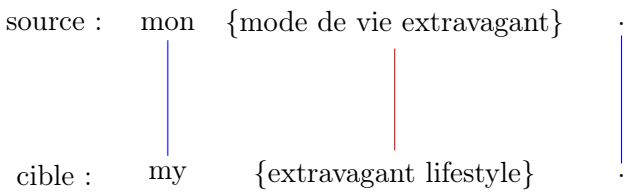


FIGURE 1.12 – Deuxième exemple d'alignement impossible dans un système de traduction à base de mots.

Ce n'est qu'au début des années 2000 que les systèmes de TAS à base de segments ont vu le

jour. Dans plusieurs travaux [Och, 2002, Zens et al., 2002, Koehn et al., 2003], des modèles de traduction basés sur l'apprentissage de paires de segments, et non pas des paires de mots, ont été proposés. Cela a permis de résoudre ou de limiter les faiblesses des systèmes à base de mots. Ces nouveaux systèmes proposaient des performances de traduction significativement meilleures que celles des systèmes à base de mots. Ces nouveaux systèmes sont devenus la référence en traduction automatique.

La construction d'une TT à base de segments se fait en deux étapes ; la première est celle de l'apprentissage statistique des alignements entre les mots, sources et cibles, en utilisant les modèles d'IBM. La deuxième étape se fait à travers des méta-heuristiques afin de détecter des paires de segments sur la base des alignements des mots. Dans la section suivante nous détaillons le processus de construction d'une table de traduction à base de segments pour la TAS.

Construction d'un modèle de traduction à base de segments

Avant d'expliquer comment est appris un modèle de traduction, à travers la construction de la table de traduction, nous devons expliquer comment est utilisé ce modèle pour l'estimation de la probabilité de traduction $P(f|e)$, celle d'avoir f comme phrase source sachant que l'hypothèse de traduction est la phrase cible e.

La probabilité $P(f|e)$ ne peut être estimée directement, car il n'existe pas de corpus d'apprentissage, contenant assez de données pour qu'il y est une répétition d'alignement entre une phrase source f et une phrase cible e. L'apprentissage est donc fait dans le but d'apprendre des probabilités de traduction entre des segments sources et cibles et non pas des phrases. De ce fait, l'estimation de la probabilité de traduction $P(f|e)$ se fait sur la base des probabilités de traduction des paires de segments qui existent entre la source f et la cible e :

$$P(f|e,a) = \prod_{i=1}^{|a|} P(S_i^f|S_{a_i}^e) \tag{1.3}$$

Dans la formule 1.3, a représente les alignements entre les segments de f et e. Considérons $S^f = \{S_1^f, S_2^f, ..., S_{|a|}^f\}$ la segmentation de la phrase source f en segments (suite de mots) et $S^e = \{S_1^e, S_2^e, ..., S_{|a|}^e\}$ la segmentation de la phrase cible e en segments aussi, où S_i^f, respectivement S_i^e, représentent une suite de mots sources, respectivement cibles. Chaque segment source S_i^f est aligné à un segment cible $S_{a_i}^e$. Chaque paire de segments alignés (S_i^f,$S_{a_i}^e$) doit exister dans la TT avec sa probabilité de traduction.

Ayant la segmentation et les alignements, la probabilité de traduction d'une hypothèse de traduction est calculée comme étant le produit des probabilités de traduction des segments alignés entre l'hypothèse et la source.

Dans la figure 1.13, nous présentons deux exemples de segmentation et d'alignement d'une même phrase source $f =$ "*la maison verte*" et son hypothèse de traduction $e =$ "*the green house*".

Pour la première segmentation, celle qui est à gauche de la figure 1.13, $S^f = \{S_1^f =$ "*la*", $S_2^f =$ "*maison*", $S_3^f =$ "*verte*"$\}$ et $S^e = \{S_1^e =$ "*the*", $S_2^e =$ "*green*", $S_3^e =$ "*house*"$\}$ avec $a^1 = \{1, 3, 2\}$. La deuxième segmentation ne contient qu'un seul segment $S^f = \{S_1^f =$ "*la maison verte*"$\}$ et $S^e = \{S_1^e =$ "*the green house*"$\}$ avec $a^2 = \{1\}$.

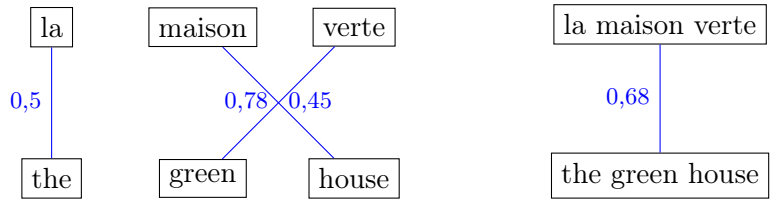


FIGURE 1.13 – Exemples de segmentations et d'alignements au niveau du segment.

En se basant sur ces deux segmentations et alignements, ainsi que sur les probabilités de traduction de chaque paire (voir Figure 1.13), nous pouvons calculer deux probabilités de traduction pour cette phrase source f et son hypothèse de traduction e :

— Segmentation 1 : $a^1 = \{1, 3, 2\}$

$$P(f|e, a^1) = P(la|the) \times P(maison|house) \times P(verte|green) = 0,5 \times 0,45 \times 0,78 = 0,175$$

— Segmentation 2 : $a^2 = \{1\}$

$$P(f|e, a^2) = P(la\ maison\ verte|the\ green\ house) = 0,68$$

Sur la base de cette explication, nous pouvons dire que la construction d'une table de traduction consiste à apprendre des alignements entre des segments sources et des segments cibles, à partir du corpus parallèle des phrases alignées, cela dans le but de les utiliser par la suite, pour la construction des traductions et l'estimation de la probabilité de traduction de chacune d'elles. L'apprentissage de ces alignements au niveau des segments se fait à travers l'apprentissage des alignements au niveau des mots, une étape que nous détaillons dans le point suivant.

Alignements au niveau des mots Le corpus d'apprentissage parallèle contient des phrases sources et cibles alignées, où dans chaque ligne nous trouvons une phrase source et sa traduction dans la langue cible. Le but à ce niveau, est de trouver, par apprentissage statistique, les alignements entre les mots de ces phrases. Ces alignements optimisent le critère de vraisemblance.

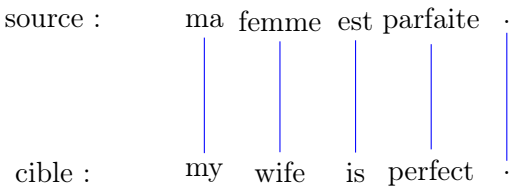


FIGURE 1.14 – Exemple d'alignement simple au niveau des mots (français-anglais).

Pour un être humain, il est facile de faire la correspondance entre les mots d'une phrase source et ceux de sa traduction et d'en trouver les alignements au niveau des mots (voir Figure 1.14). Cependant, l'automatisation de ce processus s'avère difficile et complexe, en particulier dans le cas où il y a une forte différence entre les deux langues à traiter. Comme pour le français et l'arabe où souvent un mot en arabe doit être aligné à plusieurs mots français. Pour le français et l'allemand, aussi, où il y a un fort réordonnancement entre ces deux langues (voir Figure 1.15),

les composantes des phrases (verbes, pronoms, adjectifs, etc.) n'ont pas les mêmes positions en français qu'en allemand. Les alignements que nous présentons dans la figure 1.15 peuvent être détectés par un être humain ayant des connaissances des paires de langues traitées, à l'inverse d'une machine qui nécessite un traitement plus complexe pour y parvenir.

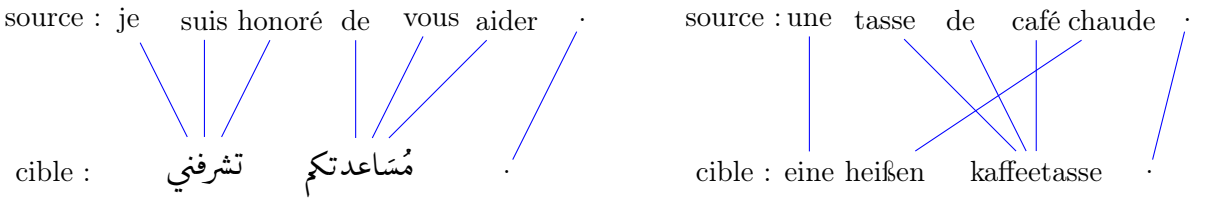


FIGURE 1.15 – Exemples d'alignements complexes au niveau des mots (français-arabe & français-allemand).

En pratique, étant donnée une traduction $e = \{e_1, e_2, \ldots, e_m\}$ de la phrase source $f = \{f_1, f_2, \ldots, f_n\}$ du corpus parallèle, où n et m représentent respectivement, la taille de f et de e en nombre de mots. La probabilité de traduction, celle d'avoir f comme phrase source de la traduction e, est estimée théoriquement, comme étant la somme de tous les alignements possibles entre les mots de f et ceux de e :

$$P(f|e) = \sum_{a \in A} P(f|e, a) \tag{1.4}$$

La taille de l'ensemble des alignements possibles (A) entre f et e au niveau des mots, croît exponentiellement en fonction de la taille de la source (n). Dans le cas d'un grand ensemble d'alignements, la somme de la Formule 1.4 ne peut être estimée dans un temps raisonnable. L'alignement final, entre les mots de la source et de la cible est alors, celui qui maximise la probabilité $P(f|e, a)$:

$$\hat{a} = \underset{a \in A}{argmax}\ P(f|e, a) \tag{1.5}$$

Sur la base de cette modélisation, les 5 modèles d'IBM assignent des probabilités d'alignement entre les mots des phrases sources et ceux des phrases cibles [Brown et al., 1990, Brown et al., 1991]. Les paramètres de ces modèles sont ajustés dans un processus itératif afin de maximiser la vraisemblance $P(f|e)$, entre les phrases sources et cibles sur le corpus d'apprentissage. Les paramètres des 5 modèles sont ajustés et optimisés d'une manière imbriquée, et passent des plus simples (IBM1) aux plus complexes (IMB5). Les paramètres du modèle IBM1 sont ajustés et utilisés par la suite pour initialiser ceux du modèle IBM2, et ainsi de suite jusqu'au modèle IBM5.

Dans le modèle IBM1, l'ordre des mots dans la phrase n'est pas pris en considération lors de l'estimation de la probabilité de traduction entre deux mots source et cible, dans ce cas, tous les alignements sont équiprobables. Dans IBM2, la position du mot dans sa phrase, est prise en compte dans l'estimation des probabilités d'alignement. Dans IBM2, les tailles de la phrase source et cible sont aussi prises en considération, ainsi que le rajout d'un mot « NULL » dans la phrase source et cible pour pouvoir aligner les mots qui n'ont pas de correspondance dans la phrase parallèle.

Dans le modèle suivant, IBM3, la notion de fertilité des mots sources est ajoutée. Un mot source peut être aligné avec plusieurs mots de la phrase cible (alignement 1-n) et le nombre de ces mots cibles représente la fertilité du mot source. Les deux derniers modèles IBM4 et IBM5 se basent sur une modélisation plus complexe dans la gestion des positions des mots dans la phrase (réordonnancement). Les alignements sont générés en plusieurs étapes par rapport aux premiers modèles.

GIZA [Al-Onaizan et al., 1999] est le logiciel le plus utilisé dans la communauté pour l'apprentissage des paramètres des modèles IBM et la génération des alignements entre mots sources et cibles à partir des phrases alignées du corpus parallèle. Ce logiciel implémente un algorithme d'alignement de mots proposé par l'équipe d'IBM [Brown et al., 1993]. La nouvelle version GIZA++ [1] [Och and Ney, 2003] implémente un algorithme qui produit les alignements de segments à partir des alignements de mots. Dans la section suivante nous présentons comment est formé un modèle de traduction à base de segments en produisant une table de traduction à base de segments.

Alignements au niveau des segments La génération des alignements au niveau des segments (suite de mots) entre une phrase source et sa cible, se fait sur la base des alignements au niveau des mots [Och, 1999, Koehn et al., 2003]. Le processus d'apprentissage statistique au niveau des mots génère une base d'alignements bidirectionnels. Dans cette base, pour chaque couple de phrases dans le corpus parallèle, nous avons deux alignements, un alignement entre les mots de la source et ceux de la cible et un autre entre les mots de la cible et ceux de la source. Dans la tableau 1.3, nous présentons un exemple d'alignement dans les deux sens entre une phrase source ($f =$ "*des décisions faciles à prendre*") et sa phrase cible ($e =$ "*easy decisions to make*") du corpus parallèle.

	easy	decisions	to	make
des				
décisions		x		
faciles	x			
à			x	
prendre				x

	easy	decisions	to	make
des		x		
décisions		x		
faciles	x			
à			x	
prendre				x

TABLE 1.3 – Exemples d'alignement bidirectionnel entre une phrase en français et sa traduction en Anglais. À gauche l'alignement français-anglais et à droite anglais-français.

La représentation matricielle des alignements est utilisée pour faciliter la détection des segments, sources et cibles, qui peuvent être alignés. En effet dans une matrice d'alignement, une case cochée représente l'existence d'un alignement entre les deux mots qui identifient cette case. Une fois les deux matrices des alignements dans les deux sens sont faites, des alignements au niveau des segments sont produits. Une partie des alignements est produite en faisant l'intersection entre les deux matrices. L'autre partie est produite en appliquant l'union des deux matrices.

Afin de produire un segment source et son segment cible aligné, les mots sources et cibles,

1. http ://www.statmt.org/moses/giza/GIZA++.html

	easy	decisions	to	make
des				
décisions		x		
faciles	x			
à			x	
prendre				x

	easy	decisions	to	make
des		x		
décisions		x		
faciles	x			
à			x	
prendre				x

TABLE 1.4 – Résultats de l'intersection (à gauche) et de l'union (à droite) des alignements entre deux phrases.

qui identifient les cases adjacentes cochées de la matrice sont regroupés. Cela nous permet de produire des segments et des alignements entre eux. Deux segments adjacents dans la matrice sont aussi regroupés pour produire des segments de tailles plus grandes. Dans les systèmes de TAS la taille maximale des segments est, souvent, fixée à 7 mots par segment, des segments plus longs ne garantissent pas de meilleures performances de traduction [Koehn et al., 2003].

Dans le Tableau 1.4, nous pouvons remarquer que l'intersection et l'union produisent des segmentations différentes avec les alignements associés au niveau des segments. Pour l'intersection (matrice de gauche), nous obtenons, de manière incrémentale comme expliqué précédemment, les segments et les alignements suivants :

— décisions – decisions
— faciles – easy
— décisions faciles – easy decisions
— à – to
— prendre – make
— à prendre – to make
— décisions faciles à prendre – easy decisions to make

L'union des deux matrices (matrice de droite) nous permet d'obtenir les alignements suivants :

— des – decisions
— décisions – decisions
— faciles – easy
— décisions faciles – easy decisions
— des décisions faciles – easy decisions
— à – to
— prendre – make
— à prendre – to make
— des décisions faciles à prendre – easy decisions to make

L'intersection permet de générer des alignements forts, dans lesquels chaque mot d'un segment source est forcément, aligné à au moins un mot du segment cible associé. L'union permet de générer des alignements dans lesquels il n'y a pas cette contrainte d'alignement entre les mots

des deux segments source et cible.

Ce processus de génération de segments et alignements entre segments est appliqué sur tous les couples de phrases alignées du corpus parallèle, afin de produire une table de traduction à base de segments [Och and Ney, 2003] avec une probabilité de traduction associée à chaque paire de segments, qui est estimée en fonction de sa fréquence d'apparition dans le corpus.

1.4.3 modèle de langue

Dans l'équation de base (Équation 1.1) des systèmes de TAS, la probabilité $P(e)$ représente le modèle de langue. Ce dernier est le deuxième modèle le plus important dans la construction d'un système de TAS après le modèle de traduction. Le modèle de langue est mis en place pour apprendre des contraintes linguistiques de la langue cible afin de guider le processus de traduction à produire des traductions syntaxiquement correctes [Koehn et al., 2003].

L'apprentissage de ces contraintes consiste à apprendre statistiquement les suites de mots syntaxiquement correctes dans la langue cible, et cela à partir d'un corpus monolingue composé de phrases dans la langue cible. Si nous supposons que le corpus d'apprentissage est constitué de phrases syntaxiquement correctes, alors les suites de mots les plus fréquentes ont plus de chances d'être correctes syntaxiquement [Brown et al., 1990]. À chaque suite de mots, est associé une vraisemblance, calculée sur la base de sa fréquence d'apparition dans le corpus, qui estime la qualité de la suite.

Au moment de la construction des traductions, le calcul de la vraisemblance $P(e)$ d'une hypothèse de traduction $e = \{e_1, e_2, \ldots, e_m\}$, de m mots, est fait comme suit :

$$P(e) = \prod_{i=1}^{m} P(e_i|e_1, ..., e_{i-1}) \tag{1.6}$$

Où e_i représente le $i^{éme}$ mot de la traduction e. La probabilité $P(e_i|e_1, \ldots, e_{i-1})$ représente la probabilité d'avoir le mot e_i dans une traduction sachant la suite de mots qui le précède. De cette manière, plus l'hypothèse de traduction est composée de suites de mots fréquentes dans la langue cible, plus elle a une meilleure probabilité d'être syntaxiquement correcte dans la langue cible.

En pratique l'apprentissage et l'estimation de telles probabilités, pour des phrases longues, s'avèrent impossible du fait du nombre de données disponibles qui ne le permettent pas. C'est pour cela que dans les modèles de langage, la taille maximale des suites de mots à prendre en considération est limitée. Ces modèles sont appelés modèles n-grammes, où n représente la taille maximale des suites de mots. Dans ce cas, l'estimation de la vraisemblance $P(e)$ est simplifiée comme suit :

$$P(e) = \prod_{i=1}^{m} P(e_i|e_{i-n+1}, ..., e_{i-1}) \tag{1.7}$$

Si un système de traduction, d'une langue source vers l'anglais, propose l'hypothèse de traduction $e =$ "*the green house*", alors la probabilité de e, selon un modèle de langue 3-grammes (voir Figure 1.10), est calculée comme suit :

— P(e)= P(the) x P(green|the) x P(house|the green) x P(.|green house)

Le plus grand problème des modèles de langage statistique est le fait que les corpus d'apprentissage ne couvrent pas tous les mots et les suites de mots d'une langue. C'est pourquoi, des techniques de lissage [Chen and Goodman, 1999] sont appliquées au cours de l'apprentissage, pour éviter d'avoir des probabilités nulles au moment de l'estimation de la probabilité du modèle de langue d'une hypothèse de traduction contenant des suites de mots inconnues du modèle. Le lissage par repli (*back-off*) [Katz, 1987] est l'une des techniques les plus utilisées dans les systèmes de traduction et que nous expliquons à travers cet exemple.

Si un 3-grammes (une suite de 3 mots dans la traduction) n'existe pas dans le modèle de langue, comme c'est le cas pour la suite *"green house ."*, nous redescendons à un modèle 2-grammes en estimant la probabilité $P(.|house)$, si ce dernier n'existe pas non plus, alors on estime la probabilité du 1-gramme $P(.)$. Cette technique est appelé le *back-off*. La probabilité de la traduction de l'exemple précédent, est calculée comme suit en utilisant les probabilités de la figure 1.10 :

— P(e)= P(the) x P(green|the) x P(house|the green) x P(.)
— P(e)= 0,5 x 0,45 x 0,38 x 0,89 = 0,076095

Si un mot de la traduction n'existe pas dans le modèle de langue, sa probabilité est remplacée par celle du mot *"unk"* qui représente tous les mots inconnus, qui n'existent pas dans le corpus d'apprentissage. Durant l'apprentissage, un seuil de fréquence d'apparition des mots, dans le corpus, est utilisé pour sélectionner les mots à prendre en compte pour la construction du modèle de langue. Les autres mots, ayant une fréquence d'apparition inférieure au seuil, sont remplacés par le mot *"unk"* qui est rajouté à l'ensemble des mots pris en compte pour l'apprentissage. De nos jours, les modèles de langage qui permettent d'atteindre les meilleures performances de traduction sont des modèles 5-grammes.

1.4.4 Décodeur

La dernière composante d'un système de traduction automatique statistique est ce qu'on appelle le décodeur. Dans le décodeur, un algorithme est implémenté pour la recherche et la construction de la meilleure traduction $\hat{e}$, pour une phrase source f en entrée.

En pratique, le décodeur prend en entrée la phrase source f et la décompose en segments sur la base des informations sauvegardées dans la table de traduction. Par la suite, le décodeur traduit chaque segment vers la langue cible, toujours en se basant sur la TT. Selon l'algorithme implémenté au niveau du décodeur, une dernière étape peut être ajoutée, dans laquelle les segments cibles sont réorganisés afin de trouver le bon ordre dans la langue cible. La segmentation et les liens entre les segments sources et cibles définissent l'alignement (a) au niveau du segment.

Au niveau du décodeur TAS, le processus de traduction est considéré comme un problème d'optimisation combinatoire difficile (Équation 1.8). En effet, pour une phrase source donnée, l'espace de recherche est énorme en raison du nombre de segmentations possibles de la source et du grand nombre de traductions possibles pour chaque segment source. Ajoutons à cette complexité, le réordonnancement (réorganisation des segments cibles) à appliquer sur la phrase cible pour construire une traduction syntaxiquement correcte. Donc, dans le décodeur, un algorithme

d'optimisation est implémenté pour construire la traduction $\hat{e}$, qui maximise la probabilité conditionnelle $P(e, a|f)$. Cette probabilité est estimée essentiellement, sur la base du score du modèle de langue $P(e)$ et du score du modèle de traduction $P(f|e, a)$.

De nos jours, les systèmes de TAS combinent avec les deux scores de base, d'autres scores estimés par d'autres modèles, qui évaluent d'autres caractéristiques de la traduction. Ces nouveaux scores sont ajoutés dans le but d'améliorer la qualité de l'évaluation des hypothèses de traduction au moment du décodage. En combinant tous les scores, le problème d'optimisation à résoudre au niveau du décodeur est reformulé comme suit :

$$\hat{e} = \underset{e,a}{argmax} \prod_{i=1}^{|h|} h_i(f, e, a)^{\lambda_i} \tag{1.8}$$

h_i est la fonction qui calcule le score du $i^{ème}$ modèle. λ_i est un poids associé à la fonction h_i. Les poids λ_i sont intégrés dans la formule, pour donner à chaque fonction h_i, un poids d'importance dans la combinaison par rapport aux autres fonctions de la combinaison.

Dans les systèmes de TAS, les scores estimés à partir des différents modèles, n'ont pas les mêmes ordres de grandeur. Och et Ney [Och and Ney, 2003] ont proposé une fonction basée sur le modèle log-linéaire, qui combine ces différents scores, pour l'évaluation des hypothèses de traduction durant le décodage. L'approche log-linéaire permet de combiner plusieurs variables, qui ne sont pas forcément dépendantes ni de même ordre de grandeur, afin de calculer un score global estimant la qualité d'une hypothèse de traduction. L'application de l'approche log-linéaire nous permet de reformuler le problème d'optimisation de la Formule 1.8 comme suit :

$$\hat{e} = \underset{e,a}{argmax} \sum_{i=1}^{|h|} \lambda_i \times log(h_i(f, e, a)) \tag{1.9}$$

Les poids λ sont des paramètres numériques à optimiser, afin de mettre en place une fonction d'évaluation guidant efficacement le processus de décodage vers la traduction optimale. Dans la section suivante, nous détaillons ce processus d'optimisation, et dans les sections d'après, nous présentons les plus importants décodeurs proposés pour la traduction automatique statistique ainsi que les plus importants algorithmes d'optimisation des paramètres λ.

1.4.5 Optimisation des paramètres

L'optimisation des paramètres λ consiste à leur trouver des valeurs qui permettent de mettre en place une fonction d'évaluation efficace. Cette fonction doit guider le décodeur dans son processus de recherche vers des traductions optimales.

L'optimisation de ces poids pour un système de TAS, se fait sur un ensemble de développement composé de n phrases sources $\{f_1, ..., f_n\}$ et de leur traduction de référence $\{r_1, ..., r_n\}$. Le but de l'optimisation est de trouver les valeurs des poids λ qui permettent de minimiser une fonction d'erreur (Err). Cette fonction d'erreur estime la différence entre les traductions des phrases sources $\{\hat{e}_1, ..., \hat{e}_n\}$, en sortie du décodeur, et les traductions de référence. Idéalement, pour effectuer cette optimisation, nous appliquons un algorithme d'optimisation classique en exécutant le décodeur plusieurs fois et en ajustant l'ensemble des poids, jusqu'à une convergence vers un

jeu de poids optimal. Ce processus d'optimisation est défini comme étant la minimisation de la somme des erreurs (différences) entre les traductions $\{\hat{e}_1, ..., \hat{e}_n\}$ et les références $\{r_1, ..., r_n\}$:

$$\hat{\lambda} = \underset{\lambda}{argmin}\Big[\sum_{i=1}^{n} Err(\hat{e}_i^{\lambda}, r_i)\Big] \tag{1.10}$$

Il existe plusieurs possibilités pour la définition de la fonction d'erreur Err, dans la littérature [Neubig and Watanabe, 2016]. Cependant, des travaux ont montré [Neubig and Watanabe, 2016], [Och, 2003] que l'utilisation d'une métrique d'évaluation de la traduction, par exemple le BLEU (Bilingual Evaluation Understudy) [Papineni et al., 2002], donne les meilleures performances d'optimisation.

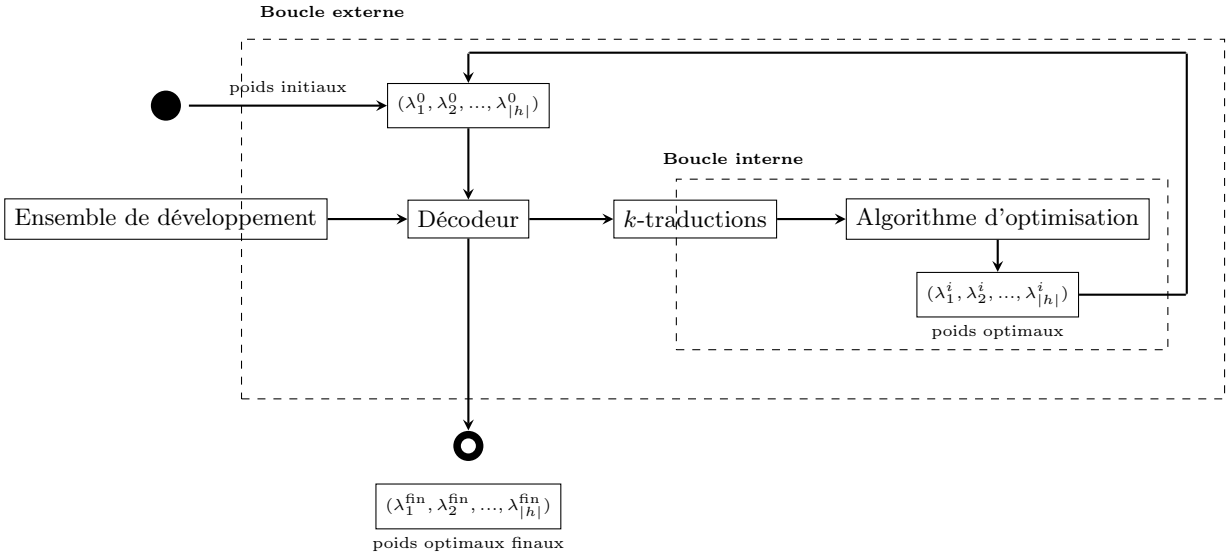


FIGURE 1.16 – Schématisation du processus d'optimisation des poids (λ).

L'exécution du décodeur à plusieurs reprises afin de trouver le jeu de poids optimal s'avère coûteuse en temps et en mémoire. C'est pour cette raison que les algorithmes d'optimisation des poids réduisent le nombre d'exécutions du décodeur.

En pratique, le processus d'optimisation des poids se déroule à travers deux processus itératifs imbriqués, que nous illustrons dans la figure 1.16. Dans le premier processus, que nous nommons "boucle externe", le décodeur utilise un jeu de poids pour traduire les n phrases sources et produit k traductions pour chaque phrase source. Dans le deuxième processus, que nous nommons "boucle interne", un algorithme d'optimisation combinatoire est appliqué pour résoudre le problème défini dans la formule 1.10.

Dans la boucle interne, l'objectif principal est de trouver le jeu de poids qui permet à la fonction d'évaluation log-linéaire de sélectionner le maximum de meilleures traductions parmi les $k \times n$ traductions des phrases sources. En effet, les k traductions de chaque source sont ordonnées selon la fonction Err. De cette manière, le jeu de poids idéal, sur l'ensemble de développement, est celui qui permet de sélectionner pour chaque phrase source sa meilleure traduction. Si une

métrique d'évaluation de traductions est définie comme fonction Err, alors le jeu de poids optimal est celui qui permet d'obtenir un ensemble de n traductions maximisant le score d'évaluation de la métrique.

Le jeu de poids optimal obtenu, en sortie de la boucle interne, est réinjecté dans la boucle externe pour relancer le décodeur et produire à nouveau k traductions pour chaque phrase source. Le processus d'optimisation global, est arrêté lorsque les poids ne peuvent plus être améliorés ou qu'aucune nouvelle traduction ne peut être produite.

Décodeurs pour la TAS

Différents décodeurs ont été proposés pour résoudre le problème d'optimisation de la Formule 1.9, celui de la production de la meilleure traduction en sortie du décodeur. Dans la communauté de la traduction automatique statistique, MOSES [Koehn et al., 2007] est l'outil de référence, et le plus utilisé, pour la construction des systèmes de TAS. Dans MOSES, un algorithme de recherche en faisceau (Beam-search) est implémenté au niveau du décodeur [Koehn, 2004]. Dans ce qui suit, nous présentons en détail le décodeur de MOSES ainsi qu'un autre décodeur utilisant un algorithme de recherche locale [Langlais et al., 2007]. Nous nous limitons à ces deux décodeurs, car ils résument deux approches de décodage. Le premier est basé sur une construction incrémentale de la traduction finale. Le deuxième décodeur se base sur la manipulation de traductions complètes tout au long du processus de recherche.

Recherche en faisceau : L'idée de base de cet algorithme est la construction incrémentale d'un ensemble de solutions (traductions complètes) en commençant avec un ensemble vide, et cela en ajoutant à chaque étape de la construction un segment cible, traduction d'un segment source non encore traduit.

La première étape de cet algorithme est de détecter et lister tous les segments possibles de la phrase source f. Ces segments sources doivent être sauvegardés dans la TT. Une hypothèse de traduction partielle est créée en traduisant un segment source détecté dans la première étape. De cette manière, plusieurs hypothèses de traduction partielles sont produites et leur nombre dépend du nombre de segments sources détectés ainsi que le nombre de traductions possibles pour chaque segment, dans la TT. Dans la figure 1.17 nous présentons une portion d'une TT pour la phrase source $f =$ "*la maison verte*". Pour cette phrase, il existe 6 segments différents et pour chaque segment source il y a 2 traductions possibles, donc pour cet exemple nous produisons 12 hypothèses de traduction partielles à la première étape de la recherche.

L'étape suivante du processus de construction des traductions consiste à faire l'extension des hypothèses partielles existantes. Pour une hypothèse partielle e^j, l'algorithme détecte tous les segments sources non encore traduits, et rajoute leur traduction à e^j. Cette manière de construire les hypothèses permet de produire incrémentalement des traductions complètes à la fin du processus de décodage. Dans l'exemple de la figure 1.18, nous pouvons voir ce processus de construction d'hypothèses de traduction complètes. À travers cet exemple, nous remarquons que les traductions complètes peuvent être produites à n'importe quel niveau de profondeur de la recherche. Nous remarquons aussi, que la taille de la phrase source, le nombre de segmentations possibles de la source ainsi que le nombre de traductions possibles de ces segments, ont un impact direct sur le nombre de traductions finales que nous pouvons produire. Ce nombre de traductions finales

la	maison	verte

the : 0,6 | house : 0,517 | green : 0,861
a : 0,2 | home : 0,196 | blue : 0,015
the house : 0,661
a home : 0,124
green house : 0,587
green home : 0,321
the green house : 0,709
the home home : 0,245

FIGURE 1.17 – Exemple d'une partie de la table de traduction concernant la phrase source *"la maison verte"* à traduire.

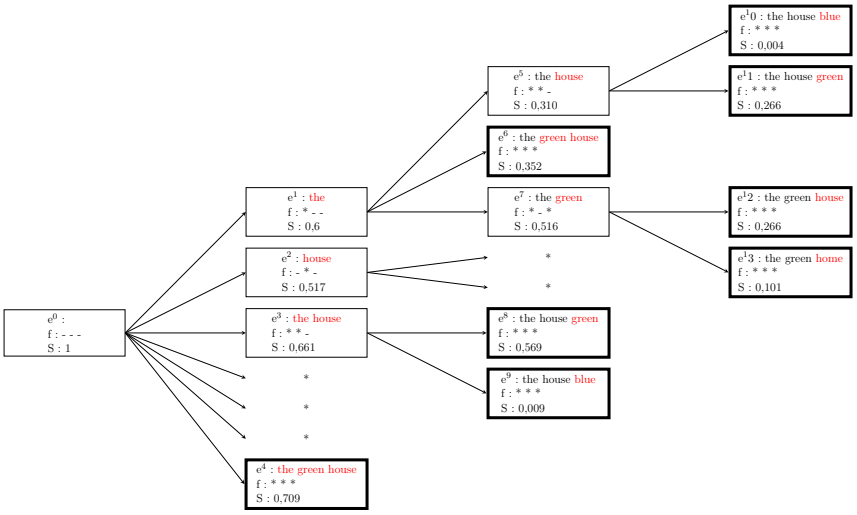


FIGURE 1.18 – Processus de l'algorithme de recherche en faisceau du décodeur dans MOSES pour une phrase source en français.

explose rapidement en fonction de ces paramètres.

Au cours du décodage, chaque hypothèse de traduction partielle ou complète est évaluée en utilisant la fonction d'évaluation suivante, qui est dérivée de la Formule 1.9 :

$$score(e^j) = \sum_{i=1}^{|h|} \lambda_i \times h_i(f, e^j, a^j) \tag{1.11}$$

De cette façon, la meilleure traduction finale $\hat{e}$ est celle qui a le score d'évaluation le plus élevé parmi les hypothèses de traduction complètes.

Dans l'exemple de décodage de la figure 1.18, nous simplifions le processus de décodage en n'utilisant que le score du modèle de traduction pour évaluer les hypothèses de traduction. En pratique, comme présenté dans la section 1.4.4, d'autres scores sont combinés pour améliorer le processus d'évaluation. Dans cet exemple, les nœuds ayant des bordures en gras représentent les traductions complètes. La traduction complète la mieux évaluée ("*the green house*" : 0.709) est donnée en sortie du système comme traduction finale.

Au cours du processus de recherche de ce décodeur, un grand nombre d'hypothèses de traduction partielles sont produites et ce nombre explose en fonction de la taille de la phrase source et de la taille de la TT. Pour éviter une explosion combinatoire, un traitement d'élagage est appliqué durant le processus de recherche. Cet élagage se base sur les scores d'évaluation des hypothèses partielles. En effet, à chaque étape du processus de construction, toutes les hypothèses de traduction partielles produites, par extension, sont placées dans des piles. Le nombre de ces piles correspond à la taille de la phrase source et dans chaque pile i, sont placées les hypothèses partielles recouvrant (traduisant) i mots de la phrase source. Dans la figure 1.19, nous illustrons cette étape de sauvegarde des hypothèses partielles dans les piles, pour l'exemple de traduction précèdent (Figure 1.18).

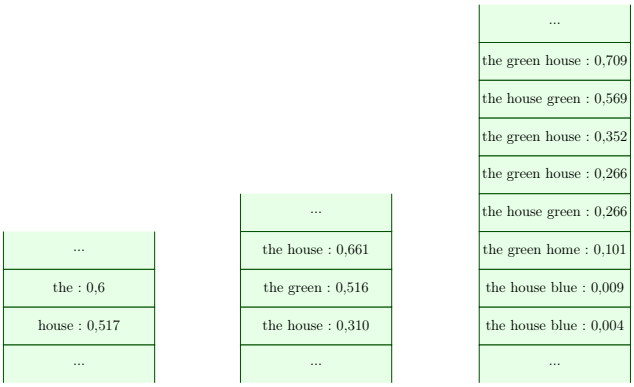


FIGURE 1.19 – Processus d'élagage dans le processus de décodage de MOSES.

Nous pouvons voir que 3 piles sont utilisées ce qui correspond aux 3 mots de la phrase source à traduire. À la fin du processus de recherche, toutes les hypothèses sauvegardées dans la $3^{ème}$ piles représentent des traductions complètes parmi lesquelles sera placée la meilleure traduction finale. Au niveau de chaque pile, les hypothèses partielles sont évaluées et triées en fonction de leur score d'évaluation et les k hypothèses les mieux évaluées, dans chaque pile, sont gardées pour être étendues dans l'étape suivante. Une autre possibilité d'appliquer l'élagage est de ne garder que les hypothèses partielles ayant des scores d'évaluation supérieurs à un seuil fixé auparavant. Dans ces deux cas d'élagage, le nombre des hypothèses à garder dans chaque pile et le seuil de score d'évaluation sont des paramètres à fixer pour assurer une bonne qualité de traduction.

Recherche locale : Langlais [Langlais et al., 2007] propose un algorithme de recherche locale comme décodeur pour la TAS. À l'inverse du décodeur de MOSES, dans ce décodeur la traduction finale n'est pas construite de manière incrémentale mais par une amélioration continue, dans un processus itératif, d'une solution complète dès le début du processus de recherche et cela en explorant son voisinage.

Le décodeur à base de recherche locale, prend en entrée la phrase source f et produit à partir de celle-ci une traduction initiale en se basant sur les informations de traduction sauvegardées dans la table de traduction. Pour produire la traduction initiale, nommée e^0, l'algorithme segmente la phrase source aléatoirement en utilisant la TT et par la suite, chaque segment source est traduit en sélectionnant l'une de ses traductions dans la TT. La concaténation des segments

cibles permet de produire une traduction complète. La traduction initiale (e^0) est évaluée en utilisant l'approche log-linéaire, tout comme MOSES, en combinant plusieurs modèles (Formule 1.9), afin de lui affecter un score.

Par la suite dans un processus de recherche itératif, l'algorithme produit à chaque itération, des traductions voisines de e^0 en appliquant des transformations au niveau du segment sur e^0. Quatre fonctions de transformation sont appliquées pour produire le voisinage [Langlais et al., 2007] :

— Le changement de la traduction d'un segment source par une autre en utilisant la TT.
— La division d'un segment source en deux segments sources adjacents, et la mise à jour des segments cibles associés en utilisant la TT.
— La fusion de deux segments sources adjacents, et la mise à jour des segments cibles associés au niveau de la cible.
— L'inversion des positions de deux segments cibles au niveau de la traduction.

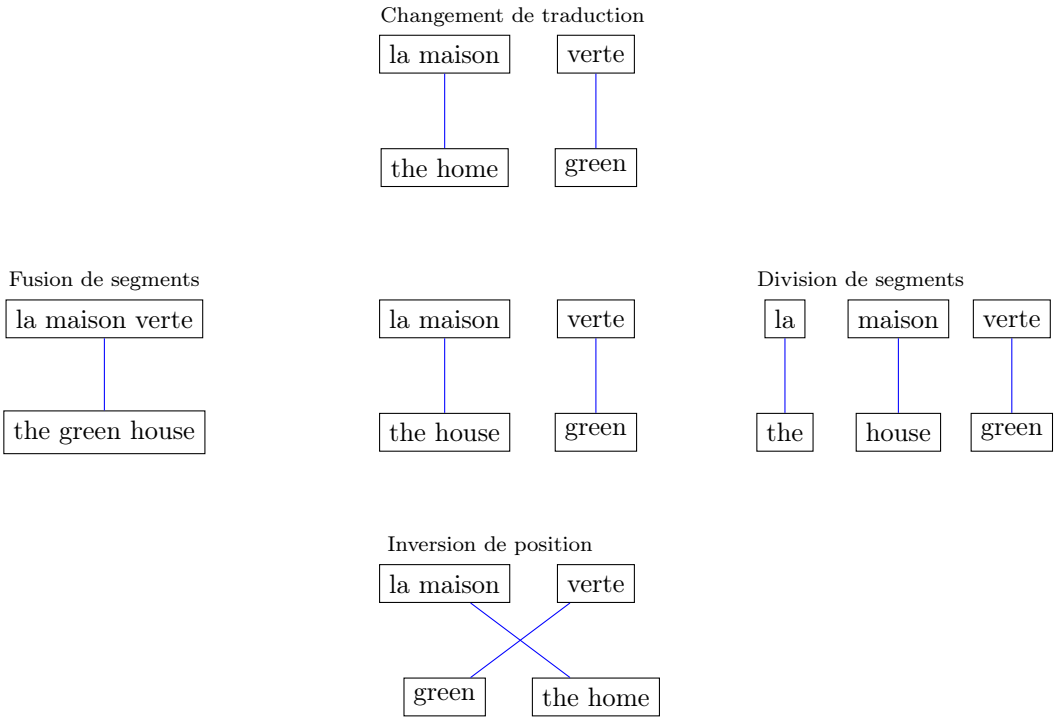


FIGURE 1.20 – Exemple de décodage avec l'algorithme de recherche locale pour une phrase en français vers l'anglais.

Dans la figure 1.20, nous illustrons le processus de génération du voisinage dans cet algorithme. Nous reprenons la même phrase source f = "*la maison verte*", présentée pour le décodeur de MOSES. À partir de la solution initiale e^0, produite aléatoirement, nous générons 4 solutions voisines (hypothèses de traduction) en appliquant les 4 fonctions de production de voisinage. D'autres solutions peuvent être générées en appliquant ces 4 fonctions sur d'autres segments de la solution e^0.

À la fin de chaque itération, toutes les traductions voisines sont évaluées. Si la traduction voisine la mieux scorée a un meilleur score d'évaluation que la traduction initiale, alors la traduction e^0 est remplacée par cette traduction voisine, sinon la traduction initiale reste inchangée. Le processus de recherche continue de la même manière itérativement, en produisant à chaque itération une traduction e^0 meilleure, ou la même, que celle de l'itération précédente. Le traitement de recherche itératif se termine dès qu'il converge vers une traduction finale qu'il ne peut plus améliorer.

Dans [Langlais et al., 2007], une autre manière d'initialisation a été testée. La traduction initiale e^0 n'est pas produite aléatoirement, mais en prenant la traduction en sortie du décodeur de MOSES comme traduction initiale. Ce test est fait pour étudier la possibilité d'améliorer la qualité de traduction produite par MOSES en utilisant le décodeur à base de recherche locale.

Comparé à l'algorithme de recherche en faisceau, implémenté dans MOSES, cet algorithme a l'avantage de manipuler des hypothèses de traduction complètes tout au long du processus de recherche. En effet, les décisions de sélection et d'élimination, prises dans cet algorithme, sont basées sur l'évaluation de traductions complètes et non pas partielles. Cela limite le risque d'éliminer de bonnes traductions au cours du processus de recherche. Dans le décodeur de MOSES, les décisions d'élagage sont prises sur la base d'évaluation de traductions partielles, où le risque d'éliminer des traductions partielles qui peuvent, par la suite, produire de bonnes traductions complètes, reste permanent.

Algorithmes d'optimisation

Plusieurs algorithmes sont proposés pour l'optimisation des paramètres d'un décodeur dans un système de TAS [Neubig and Watanabe, 2016]. Comme nous l'avons présenté dans la section 1.4.5, ces algorithmes se déroulent en deux boucles imbriquées : la boucle externe, pour le lancement du décodeur à optimiser, afin de traduire l'ensemble de développement ; et la boucle interne, dans laquelle ces algorithmes d'optimisation sont implémentés.

Dans ce qui suit, nous présentons MERT [Och, 2003] l'algorithme d'optimisation de référence et deux autres algorithmes proposés afin de résoudre le problème d'optimisation des poids λ au niveau de la boucle interne.

MERT : Minimum Error Rate Training [Och, 2003] est considéré comme l'algorithme d'optimisation de référence. Il est implémenté dans MOSES afin d'optimiser son décodeur [Koehn et al., 2007]. Dans MERT, un algorithme de recherche linéaire est implémenté au niveau de la boucle interne dans le but de trouver le meilleur jeu de poids. Dans cet algorithme, la valeur de chaque poids est optimisée séparément en fixant les valeurs des autres poids. La valeur d'un poids λ_i est optimisée linéairement en minimisant la Formule 1.10 avec une valeur ($alpha$) de déplacement dans l'espace de recherche. Une fois tous les poids optimisés séparément, le même processus d'optimisation linéaire est réitéré jusqu'à une convergence vers un jeu de poids optimal. Le jeu de poids obtenu dans la boucle interne est injecté dans la boucle externe pour retraduire l'ensemble de développement.

Dans MERT [Och, 2003], la fonction Err est définie par BLEU [Papineni et al., 2002], la métrique d'évaluation des traductions de référence dans la communauté. Donc, le jeu de poids

optimal doit maximiser le score BLEU des traductions produites par le décodeur sur l'ensemble de développement.

PRO : Dans PRO (Pairwise Ranking Optimization) [Hopkins and May, 2011], le processus d'optimisation est considéré comme un problème de classification. Dans la boucle externe, le décodeur traduit l'ensemble de développement et produit pour chaque phrase source k traductions en sortie. Au niveau de la boucle interne, le processus d'optimisation consiste à trouver le jeu de poids qui arrive à trier les k traductions de chaque phrase source de la même manière que le classement de ces traductions en utilisant une métrique d'évaluation de la traduction (BLEU).

PRO, par rapport à d'autres algorithmes d'optimisation, nécessite une étape d'apprentissage pour apprendre à trier les k hypothèses de traduction. Une fonction de comparaison deux-à-deux est apprise pour être utilisée par la suite au moment de l'optimisation des poids.

PSO : Dans cet algorithme, l'optimisation des poids au niveau de la boucle interne est faite à l'aide d'un algorithme d'optimisation par essaims particulaires [Kocur and Bojar, 2016] (Particle Swarm Optimization), l'un des algorithmes bio-inspirés [Binitha and Sathya, 2012] les plus utilisés en optimisation combinatoire. Le processus d'optimisation simule le comportement d'un groupe d'oiseaux dans le but d'arriver à la bonne source de nourriture.

En pratique, dans une population d'agents, chacun d'entre eux représente un jeu de poids et se déplace dans l'espace de recherche. Les déplacements des agents sont guidés par un certain nombre de règles qui simulent le mécanisme de communication entre les agents. Dans l'algorithme chaque agent est représenté par quatre éléments, sa position actuelle dans l'espace de recherche (solution actuelle), la meilleure solution qu'il a trouvée depuis le début, la meilleure solution trouvée par son voisinage depuis le début, et finalement sa vitesse de déplacement. Ces éléments vont influencer les déplacements futurs de chaque agent. En effet, à chaque itération, un agent ajuste sa vitesse de déplacement dans l'espace de recherche et sa position en fonction de la meilleure solution qu'il a trouvée et de la meilleure solution du voisinage. Le processus d'optimisation continue jusqu'à une convergence vers un jeu de poids optimal.

À fin d'évaluer les solutions au cours de ce processus de recherche, la métrique BLEU est utilisée par défaut pour estimer la qualité des traductions sélectionnées en utilisant le jeu de poids représenté dans chaque solution.

Dans [Neubig and Watanabe, 2016], une panoplie d'algorithmes d'optimisation pour la traduction automatique sont présentés. Les résultats présentés dans ce papier montrent que MERT reste l'algorithme le plus utilisé pour l'optimisation des paramètres des systèmes de traduction statistique. Cependant, MERT a une faiblesse majeure : les performances d'optimisation se dégradent en fonction du nombre de paramètres à optimiser [Hopkins and May, 2011].

1.5 Évaluation de la qualité de la traduction

Les différents systèmes de TA, peuvent proposer des traductions différentes pour une même phrase source. Cette différence de traduction peut s'expliquer par l'approche sur laquelle se base chaque système ainsi que par les algorithmes d'apprentissage et de décodage implémentés dans chaque système. Cependant, cela ne nous informe en rien sur les performances de ces systèmes

et sur la qualité des traductions qu'ils produisent.

L'évaluation de la qualité de la traduction constitue, à elle seule, un important domaine de recherche en TA. L'évaluation des traductions fait appel à d'autres techniques et approches en traitement automatique des langues que celles présentées pour la TA. Évaluer un système de traduction revient à évaluer les traductions qu'il produit. Cette évaluation se fait généralement sur un corpus parallèle de test contenant des paires de phrases différentes de celles des corpus d'apprentissage et de développement. Le système de traduction à évaluer traduit toutes les phrases sources du corpus de test et par la suite, l'évaluation se fait par l'estimation du degré de similarité entre les traductions produites et les traductions de référence du corpus de test. À chaque traduction, un score est calculé, estimant sa qualité. Un score global est calculé pour l'ensemble des traductions, afin d'estimer la qualité des traductions au niveau du corpus. Cette manière de faire, nous permet d'évaluer la qualité d'un système de traduction et de le comparer à d'autres systèmes. Plusieurs campagnes d'évaluation des systèmes de traduction sont régulièrement organisées à l'image de WMT [Bojar et al., 2017], où plusieurs équipes de recherche proposent leurs systèmes de traduction et les comparent en utilisant les mêmes données d'apprentissage, de développement et de test.

Les premiers travaux dans ce domaine, étaient basés sur une évaluation humaine des traductions [Han and Wong, 2016]. Dans une évaluation humaine, la qualité d'une traduction est estimée en la comparant soit à la phrase source, soit à la traduction de référence soit aux deux en même temps. L'évolution de la TA, et le nombre croissant de systèmes qui lui sont consacrés, ainsi que la grande quantité de données utilisées pour l'apprentissage et le test, ont rendu l'évaluation humaine coûteuse en temps et en argent. Cela a mené à l'automatisation de ce processus d'évaluation. En effet, de nos jours, plusieurs techniques d'évaluation automatique sont disponibles [Han and Wong, 2016]. Ces métriques d'évaluation automatique se basent généralement sur une comparaison lexicale entre les traductions produites par les systèmes de traduction et les traductions de référence. D'autres techniques proposent des métriques d'évaluation hybridant évaluation humaine et automatique.

Il existe plusieurs défis dans la mise en place d'un processus automatique d'évaluation de la traduction. Le premier défi est la disponibilité de traductions de référence de bonne qualité et pouvant être atteignables par un système de traduction. En effet, les traductions de référence sont produites par des experts humains qui, dans certains cas, traduisent le sens d'une phrase dans son contexte, un contexte qui ne peut être pris en compte par un système de traduction automatique. Si nous prenons l'exemple de construction des corpus de traduction des débats du parlement européen (Europarl [Bojar et al., 2017]), les paroles d'un intervenant sont transcrites dans sa propre langue par un expert humain, et en même temps d'autres experts, traduisent ces paroles vers d'autres langues selon le contexte du débat en cours. Le fait d'avoir recours à différents experts humains peut générer une grande différence entre les phrases sources et leur traduction de référence, en particulier entre les langues peu utilisées dans les débats. Dans l'exemple du Tableau 1.5, nous pouvons voir un exemple de phrases parallèles issues d'un corpus Europarl (français-anglais). La traduction de référence en anglais est quasiment impossible à atteindre par un système de TA en se basant sur la source. Dans ce cas, si nous comparons la traduction produite par un système (Google Translate) avec la référence, la qualité de traduction sera mauvaise même si cette traduction reproduit bien le sens de la phrase source.

Ce premier problème, peut être géré par une évaluation humaine qui prend en considération

Phrase source	Traduction de référence	Traduction de Google Translate
il est superflu de souligner la difficulté et la complexité de la réunification allemande .	here is no need for me to emphasize how difficult and complex german reunification was in its entirety .	it is superfluous to underline the difficulty and complexity of german reunification .

TABLE 1.5 – Exemple de phrases parallèles (français-anglais) avec une traduction de référence (anglais) non atteignable.

la phrase source, et non pas que la référence, pour estimer la qualité de traduction. Cependant, le recours fréquent à l'expertise humaine pose d'autres problèmes, en particulier, celui de la reproductibilité d'un même processus d'évaluation sur un grand ensemble de test et pour plusieurs systèmes de traduction. La subjectivité de l'évaluation humaine rend cette reproductibilité impossible.

Contrairement à l'évaluation humaine, les métriques d'évaluation automatique assurent la reproductibilité d'un même processus d'évaluation, mais ne prennent pas en considération la phrase source et son contexte dans l'évaluation. Dans certains corpus, plusieurs traductions de référence sont disponibles pour une même phrase source. Comparer une traduction d'un système avec différentes traductions de référence dans un processus d'évaluation, minimise l'impact de la qualité des traductions de référence.

Dans les points suivants, nous présentons brièvement une des techniques d'évaluation humaine et deux métriques d'évaluation automatique largement utilisées dans les travaux de recherche en TA. Ces deux métriques automatiques sont utilisées dans ce travail de thèse pour l'évaluation des performances de notre système de traduction.

1.5.1 Évaluation humaine

Dans cette approche, des experts humains se partagent l'ensemble des traductions produites par le système de traduction. Chacun des experts évalue les traductions qui lui sont affectées, selon des critères prédéfinis à l'avance. Pour chaque critère, l'expert assigne un score à chaque traduction, ce score est délimité par une valeur maximale et une valeur minimale. Plus le score est élevé, meilleure est la traduction selon ce critère.

Les critères d'évaluation varient selon ce que l'on veut évaluer. Parmi ces critères, il y en a deux souvent utilisés dans les campagnes d'évaluation des systèmes de traduction [Han and Wong, 2016] :

— **A-score (Adequacy)**, qui estime à quel point la traduction arrive à transcrire le sens de la phrase source. La construction syntaxique de la traduction n'est pas prise en considération dans ce processus d'évaluation.

— **F-score (Fluency)**, qui estime à quel point la traduction générée est grammaticalement correcte dans la langue cible. Dans ce cas, l'expert ne prend pas en compte le sens de la traduction mais juste sa structure.

Ces deux scores sont combinés, éventuellement, avec d'autres scores évaluant d'autres critères, afin de donner un score final estimant la qualité de chaque traduction.

Ces évaluations manuelles ont l'avantage d'être réalisées par des experts humains ce qui permet de prendre en considération la phrase source et son contexte. Cela donne au résultat de l'évaluation un certain degré de confiance d'un point de vue humain. Cependant, cette évaluation reste fortement subjective et quasiment impossible à reproduire afin d'évaluer plusieurs systèmes de la même manière. Rajoutons à cela le fait que l'évaluation humaine est coûteuse en temps et en argent.

1.5.2 Évaluation automatique

L'évaluation automatique de la qualité de la traduction vise à réduire le coût et le temps du traitement d'évaluation ainsi qu'à rendre ce processus d'évaluation reproductible. Cette reproductibilité nous permet de comparer plusieurs systèmes, ou plusieurs versions d'un même système à des moments différents, et cela de manière objective.

Toutes les métriques d'évaluation automatique se basent sur la comparaison des hypothèses de traduction produites par le système de TA, avec les traductions de référence produites par des experts humains. Une métrique d'évaluation assigne un score à chaque traduction ou à l'ensemble du corpus de test. Ce score estime le degré de similarité lexicale entre les hypothèses et les références. D'autres métriques utilisent plus de ressources afin d'améliorer le processus d'évaluation, comme l'analyse syntaxique de la traduction ainsi que des dictionnaires de synonymes [Han and Wong, 2016].

Dans ce qui suit, nous présentons deux métriques d'évaluation automatique souvent utilisées dans le domaine de la TA pour l'évaluation des systèmes de traduction, à savoir le BLEU [Papineni et al., 2002] et le TER [Snover et al., 2006]. Dans [Han and Wong, 2016] nous trouvons d'autres métriques d'évaluation automatique telle que METEOR [Banerjee and Lavie, 2005] et WER [Su et al., 1992], ces dernières ont, respectivement, le même principe d'évaluation que celui de BLEU et TER et peuvent être considérées comme leurs variantes.

BLEU

La métrique automatique la plus utilisée dans le domaine de la traduction automatique pour l'évaluation des sorties des systèmes est le BLEU (Bilingual Evaluation Understudy) proposée par Papineni [Papineni et al., 2002] en 2002. Cette métrique estime la similarité lexicale entre les hypothèses de traduction et les traductions de référence au niveau du corpus.

Le score calculé par le BLEU est considéré comme une valeur de précision des sorties du système par rapport aux références. Le calcul de ce score se base sur le nombre de recouvrements des n-grammes (segments de n mots) qui existent entre les hypothèses de traduction et les références. Dans [Papineni et al., 2002], la valeur de n varie entre 1 et 4 pour évaluer ce recouvrement sur des segments d'un, deux, trois et quatre mots. Le score BLEU final est calculé comme suit :

$$BLEU = BP \times exp(\sum_{n=1}^{N} w_n \; log \; Precision_n) \tag{1.12}$$

$$BP = \begin{cases} 1 & \text{if } c > r, \\ e^{1-\frac{r}{c}} & \text{else.} \end{cases} \tag{1.13}$$

$Precision_n$ est la valeur de précision de recouvrement entre les segments de taille n, des traductions du système et ceux des références. w_n est un poids associé à chaque ensemble de segment de taille n. Ces poids sont utilisés pour donner de l'importance à des valeurs de précision de certains n-grammes par rapport aux autres. Les 4-grammes par exemple, peuvent avoir des valeurs de précision nulles, en particulier dans le cas des courtes phrases, dans ce cas si un segment de taille 4 existe dans l'hypothèse et la référence sa valeur de précision doit être valorisée en ajustant le poids w_4. BP est une pénalité de brièveté qui estime la différence entre la somme des tailles des hypothèses de traduction (c) et la somme des tailles des traductions de référence (r). Cette pénalité est ajoutée au calcul du score pour ne pas favoriser les systèmes qui produisent des traductions courtes. En effet, les traductions courtes ont de bonnes valeurs de précision vu le peu de segments à comparer avec les références. Cette pénalité réajuste le score BLEU final, proportionnellement à la différence entre les tailles des traductions et des références et n'a aucun impact sur le calcul dès que la somme des tailles des traductions dépasse celle des tailles des références.

TER

TER (Translation Edit Rate) est une métrique estimant aussi la similarité lexicale entre l'hypothèse et la référence [Han and Wong, 2016]. Le TER calcule automatiquement le nombre de modifications (éditions) à appliquer sur l'hypothèse de traduction afin d'atteindre la traduction de référence. Le score TER d'une traduction e est calculé comme présenté dans la Formule 1.14, où le score de chaque traduction correspond au nombre d'éditions à lui appliquer divisé par la longueur, en nombre de mots, de la traduction de référence. La qualité de la traduction du corpus de test est estimée en calculant une moyenne des scores TER de toutes les traductions.

$$TER(e) = \frac{\#\text{editions}}{\#\text{mots de la référence}} \tag{1.14}$$

Quatre types d'édition à appliquer sur l'hypothèse de traduction sont prises en compte dans le calcul du score TER [Snover et al., 2006] :

— La substitution d'un mot par un autre.
— L'insertion d'un mot.
— La suppression d'un mot.
— le déplacement (décalage) d'un mot ou d'une suite de mots successifs dans la traduction.

Comparé à la métrique BLEU, TER est souvent utilisé pour pouvoir estimer la qualité des traductions au niveau de la phrase. Le score BLEU estime la qualité des traductions au niveau du corpus.

le HTER (Humain Translation Edit Rate) [Snover et al., 2006] est une métrique d'évaluation hybride, qui nécessite une expertise humaine suivie d'une évaluation automatique. Dans le HTER, l'expert humain corrige manuellement les traductions du système, soit en se référant aux traductions de référence soit sans aucune information autre que les traductions du système. Par la suite le score TER est calculé pour les traductions du système en les comparant avec les traductions corrigées par l'expert. Cette manière de faire, assure une rapidité dans l'intervention humaine, vu l'utilisation du TER. D'un autre côté, elle permet de bien évaluer les traductions correctes, d'un point de vue humain, même si elles sont différentes des traductions de référence.

1.6 Conclusion et discussion

Dans ce chapitre, nous avons présenté les principes de base de la traduction automatique ainsi que les plus importantes techniques et approches proposées dans le domaine. Nous avons présenté plus en détail l'approche statistique à base de segments pour la traduction, en abordant ses principes de base, ses principales composantes et la manière dont est construit un système de traduction statistique à base de segments.

Une des composantes d'un système de traduction statistique à base de segments, que nous avons présentée dans ce premier chapitre, est le décodeur. Au niveau du décodeur, un algorithme d'optimisation combinatoire est implémenté pour construire et trouver la traduction optimale pour une phrase source en entrée. Nous avons présenté deux décodeurs différents qui utilisent deux algorithmes d'optimisation différents. Le décodeur de référence, celui de MOSES est basé sur une construction incrémentale des traductions en utilisant un algorithme de recherche en faisceau avec élagage. Le deuxième décodeur, utilise une méta-heuristique, algorithme de recherche locale, afin de trouver la traduction optimale. Ces deux décodeurs n'explorent pas la totalité de l'espace de recherche, chose qui est impossible en un temps raisonnable, donc le risque de passer à côté de la meilleure traduction existe dans les deux cas. Ce risque est inévitable dans le cadre de la traduction statistique, mais une bonne stratégie d'exploration de l'espace de recherche ainsi que de bonnes stratégies de prise de décisions peuvent maximiser les chances de trouver la meilleure traduction.

Le décodeur de MOSES, explore l'espace de recherche de manière presque exhaustive, ce qui lui permet de tester un grand nombre d'hypothèses de traduction, parmi lesquelles il peut exister la traduction optimale. Cependant, l'élagage appliqué au cours du processus de recherche afin d'éviter l'explosion combinatoire, dans lequel les décisions sont prises sur la base d'évaluations d'hypothèses partielles et non complètes, peut provoquer l'élimination d'hypothèses partielles prometteuses. L'extension de ces hypothèses aurait pu produire de bonnes traductions finales. Le décodeur à base de recherche locale a l'avantage de manipuler des hypothèses de traduction complètes. Cela lui permet de prendre les bonnes décisions dans la sauvegarde ou l'élimination d'une solution. Cependant, le processus d'exploration de l'espace de recherche de cet algorithme est aléatoire et fortement dépendant de la solution initiale, qui est elle-même générée de manière aléatoire. L'autre risque dans cet algorithme, est la convergence rapide vers un optimum local si les solutions voisines ne permettent pas de se diriger vers la traduction optimale globale, sachant que le déplacement dans l'espace de recherche dans un algorithme de recherche locale est faible et ne permet pas un déplacement radical pour éviter les optimums locaux.

Dans ce travail de thèse, nous étudions l'utilisation des algorithmes bio-inspirés, principalement les algorithmes génétiques (AG), comme décodeur pour la traduction automatique statistique à base de segments. Dans la littérature des algorithmes d'optimisation, les algorithmes génétiques sont connus pour leur efficacité d'exploration et d'exploitation d'un grand espace de recherche. L'application de ces algorithmes au problème de décodage peut nous permettre de manipuler des hypothèses de traduction complètes tout au long du processus de recherche, à l'inverse du décodeur de MOSES. Ils permettent aussi, de se déplacer de manière plus efficace dans l'espace de recherche, si nous le comparons à un algorithme de recherche locale, car ils n'utilisent pas qu'une solution à la fois, mais une population de solutions.

En plus du décodeur, nous utilisons aussi les algorithmes génétiques pour résoudre le problème de l'optimisation des poids de la fonction log-linéaire, utilisée pour évaluer la qualité des hypothèses de traduction au cours du décodage. En effet, dans cette thèse nous proposons des implémentations d'algorithmes génétiques pour résoudre les principaux problèmes d'optimisation au niveau du décodeur d'un système de traduction statistique à base de segments.

Dans le chapitre suivant, nous présentons de manière globale les algorithmes d'optimisation bio-inspirés et leurs principales familles. Par la suite, nous présentons plus en détail les algorithmes génétiques et leur fonctionnement. Ainsi que les plus importants travaux de recherche utilisant les algorithmes génétiques pour la traduction automatique afin de situer nos contributions.

2

Les Algorithmes Bio-inspirés

2.1 Introduction

En traduction automatique par ordinateur, particulièrement en traduction automatique statistique, différentes composantes des systèmes sont considérées comme étant des problèmes d'optimisation. En effet, pour la construction d'un système de traduction statistique (voir Section 1.4.1 du chapitre précèdent), l'apprentissage du modèle de traduction, le décodage et l'optimisation des poids la fonction log-linéaire sont trois importants problèmes d'optimisation. Savoir conceptualiser et définir le bon algorithme pour la résolution de ces problèmes d'optimisation est une tâche importante afin de mettre en place un système de traduction performant.

Les problèmes d'optimisation sont des problèmes mathématiques que nous rencontrons dans différentes disciplines en informatique et en ingénierie. En effet, l'optimisation consiste à trouver la solution optimale pour un problème donné. Cette solution optimale est dite atteignable dans le cas des problèmes simples, pour lesquels il peut y avoir un algorithme déterministe permettant de visiter la totalité de l'espace de recherche, en un temps raisonnable, et de vérifier toutes les solutions possibles pour y trouver la solution optimale [Papadimitriou and Steiglitz, 1998]. À titre d'exemple, rechercher les solutions d'une équation du second degré est un problème d'optimisation pour lequel il existe un algorithme déterministe (voir Figure 2.1).

$$\text{Problème : } a.x^2 + b.x + c = 0$$

$$\text{Solution : } \begin{cases} \Delta = 2b - 4ac & \\ Pas\ de\ solution & \Delta < 0, \\ x = \frac{-b}{2a} & \Delta = 0, \\ x_1 = \frac{-b-\sqrt{\Delta}}{2a}\ et\ x_1 = \frac{-b+\sqrt{\Delta}}{2a} & \Delta > 0. \end{cases}$$

FIGURE 2.1 – Exemple 1 de problème d'optimisation facile : équation de 2^e degré.

Un autre exemple de problème ayant une solution atteignable est le problème de satisfaisabilité booléenne (SAT) [Zhang et al., 2001] à 3 variables booléennes (x_1, x_2, x_3) et 5 clauses logiques $(c_1, c_2, c_3, c_4, c_5)$. SAT est considéré comme le problème NP-complet de référence [Zhang et al., 2001] dans lequel le but est d'affecter une valeur booléenne à chaque variable x_i de façon à satisfaire le maximum de clauses logiques. Pour un problème SAT à 3 variables, l'espace de recherche est

composé de $2^3 = 8$ solutions possibles. Dans ce cas, un algorithme de recherche exhaustive peut facilement explorer la totalité de l'espace de recherche et vérifier toutes les solutions (8) pour trouver la solution optimale, en un temps raisonnable (voir Figure 2.2).

clauses :
$c_1 = x_1 \vee x_2$
$c_2 = x_1 \vee \neg x_3$
$c_3 = \neg x_1 \vee x_2 \vee \neg x_3$
$c_4 = x_2 \vee \neg x_3$
$c_5 = \neg x_1 \vee \neg x_2 \vee \neg x_3$

Solution : $x_1 = 1,\ x_2 = 0,\ x_3 = 0$

FIGURE 2.2 – Exemple 2 de problème d'optimisation facile : problème SAT à 3 variables et 10 clauses.

D'autres types de problèmes d'optimisation, ayant un très grand espace de recherche et même une infinité de solutions pour certains problèmes, sont plus complexes. Dans ces cas-là, nous ne pouvons pas définir un algorithme déterministe capable d'explorer la totalité de l'espace de recherche et d'y trouver la solution optimale en un temps de calcul raisonnable [Papadimitriou and Steiglitz, 1998]. Un problème SAT à 100 variables et 1 million de clauses logiques à satisfaire est complexe car l'espace de recherche est composé de $2^{100} = 1.2676506 \times 10^{30}$ solutions possibles. Dans ce cas, l'exploration de l'espace de recherche de manière exhaustive en un temps raisonnable est quasiment impossible. L'optimisation des paramètres d'un système en ingénierie est un problème ayant un espace de recherche infini, en particulier dans le cas où les paramètres sont des variables à valeurs réelles. Pour cette catégorie de problèmes d'optimisation, des méta-heuristiques sont utilisées pour explorer le plus efficacement possible, de manière stochastique et non exhaustive, l'espace de recherche afin de se rapprocher de la solution optimale.

Les algorithmes bio-inspirés sont des méta-heuristiques stochastiques, pour l'optimisation, inspirées de différents phénomènes du monde naturel. Ces algorithmes s'inspirent particulièrement des comportements sociaux des animaux et insectes, les évolutions des espèces ou les comportements intelligents de certains organismes. Dans la nature, ces espèces et organismes arrivent toujours à adopter la bonne stratégie (comportement) pour se défendre (ex : système immunitaire), pour trouver de la nourriture (ex : fourmis et abeilles) ou s'adapter à un environnement (ex : évolution génétique). Ces différentes tâches de la vie courante, dans la nature, peuvent être considérées comme des problèmes d'optimisation complexes, vu la difficulté à les résoudre et l'énorme nombre de possibilités de solutions que nous pouvons trouver dans la nature. Les stratégies adoptées par les différentes espèces, pour résoudre ces problèmes et pour arriver à certains objectifs, peuvent être considérées, quant à elles, comme des algorithmes d'optimisation.

La diversité de ces phénomènes dans la nature a été, et l'est toujours, une riche source d'inspiration pour les scientifiques qui ont proposé un grand nombre d'algorithmes bio-inspirés pour l'optimisation [Binitha and Sathya, 2012]. Ces algorithmes sont basés généralement, sur une amélioration itérative d'une solution (ex : recherche taboue) [Glover, 1989] ou d'une population de solutions (ex : évolution génétique) [Črepinšek et al., 2013]. Cette amélioration se fait à travers un processus itératif dans lequel de nouvelles solutions, idéalement, meilleures que les précédentes, sont produites. Le comportement de ces algorithmes est souvent aléatoire guidé par une fonction objectif qui évalue la qualité des solutions tout au long du processus de recherche.

FIGURE 2.3 – Taxonomie des différents algorithmes bio-inspirés pour l'optimisation.

Cette fonction objectif permet d'orienter le processus d'exploration vers la solution optimale ou de s'en rapprocher.

Dans le point suivant, nous présentons les différentes catégories d'algorithmes d'optimisation bio-inspirés. Les algorithmes génétiques sont beaucoup plus détaillés dans la section suivante.

2.2 Les catégories des algorithmes bio-inspirés

Une panoplie de méta-heuristiques bio-inspirées existe dans la littérature. Ces algorithmes peuvent être classés en trois grandes catégories, à savoir les algorithmes évolutionnaires, les algorithmes à base d'intelligence distribuée et les algorithmes inspirés de l'écologie. Binitha et Sathya [Binitha and Sathya, 2012] illustrent cette catégorisation, dans une taxonomie que nous

présentons dans la figure 2.3. Cette taxonomie liste les différents algorithmes bio-inspirés pour l'optimisation.

Dans ce qui suit, nous détaillons les principes de ces trois catégories d'algorithmes bio-inspirés tout en citant quelques exemples d'algorithmes pour chaque catégorie.

2.2.1 Algorithmes évolutionnaires

Les algorithmes évolutionnaires sont considérés comme les premiers algorithmes bio-inspirés en intelligence artificielle pour l'apprentissage automatique et l'optimisation combinatoire. Ces algorithmes sont inspirés de la théorie de l'évolution des espèces naturelles [Back, 1996]. Comme pour l'évolution des espèces, le principe de base de ces algorithmes est l'évolution d'une population d'individus (solutions), dans le temps et d'une génération à une autre, vers une population d'individus mieux adaptés à leur environnement. Donc, une évolution vers des solutions meilleures pour le problème d'optimisation à résoudre. Ces algorithmes ont un comportement stochastique, dans lequel le processus d'optimisation se déroule de manière itérative avec une prise de décision souvent aléatoire, mais guidée par une fonction d'évaluation des individus appelée fonction objectif ou *fitness* [Back, 1996].

Selon la théorie de l'évolution des espèces [Holland, 1973], plusieurs mécanismes sont impliqués dans l'évolution d'une population d'individus. Le premier mécanisme est celui de l'encodage des caractéristiques des individus, où les caractéristiques de chaque individu de la population sont sauvegardées dans ses gènes. La qualité d'un individu ou son degré d'adaptation à son environnement est estimée en fonction de ses caractéristiques. Les caractéristiques des différents individus se transmettent d'une génération à une autre à travers des mécanismes de reproduction et de transformation au niveau des gènes. Un mécanisme de sélection est appliqué naturellement, où les individus les mieux adaptés ont plus de chances de se reproduire afin de transmettre leurs caractéristiques aux générations suivantes. La fin de chaque génération est marquée par un mécanisme de remplacement dans lequel les individus les mieux adaptés sont gardés et les autres éliminés pour assurer la survie de l'espèce. Ces différents mécanismes assurent, dans le monde naturel, une évolution continue d'une espèce vers des individus toujours mieux adaptés à leur environnement afin d'assurer sa survie.

Les algorithmes évolutionnaires [Fogel, 2006], [Črepinšek et al., 2013] simulent ce processus naturel de l'évolution en adaptant les différents mécanismes naturels au problème d'optimisation en question. La première étape de cette adaptation, consiste à trouver la bonne représentation des solutions du problème. Dans cette représentation, l'unité de base d'une solution est sauvegardée dans des gènes. Par la suite, il faut définir une fonction d'évaluation ou fonction *fitness*, qui permet d'évaluer la qualité d'une solution par rapport au problème à traiter. Une fois ces deux mécanismes adaptés au problème, il faut définir des stratégies de sélection, de remplacement ainsi que celle de la reproduction pour générer de nouvelles solutions à partir de la population existante.

Fogel synthétise le fonctionnement basique des algorithmes évolutionnaires [Fogel, 2006] dans un schéma que nous présentons dans la figure 2.4. Toutes les variantes des algorithmes évolutionnaires se basent sur ce déroulement basique (Figure 2.4) dans lequel la principale idée est de faire évoluer une population de solutions vers des solutions meilleures en produisant de nouvelles solutions.

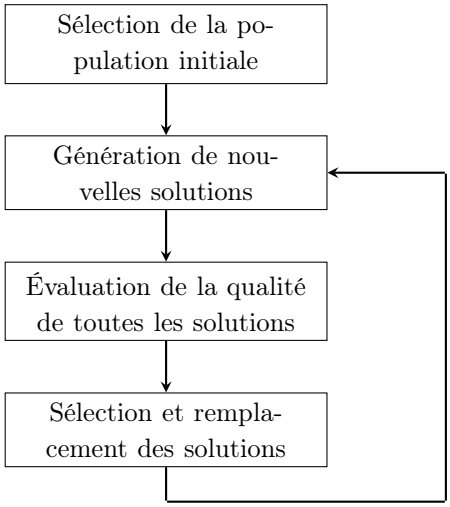


FIGURE 2.4 – Le déroulement basique des algorithmes évolutionnaires.

Depuis les années 60, plusieurs algorithmes évolutionnaires ont été proposés [Binitha and Sathya, 2012]. Parmi les différents algorithmes évolutionnaires qui existent dans la littérature, trois grandes familles se distinguent en particulier. Nous présentons ces trois familles dans les points qui vont suivre.

Stratégie d'évolution : cet algorithme a été proposé en 1964 [Beyer and Schwefel, 2002] par un groupe d'étudiants dans le but d'optimiser la conception d'un problème aérodynamique. Cet algorithme qui a été nommé *"Stratégie d'évolution"* est considéré comme l'une des premières versions des algorithmes évolutionnaires. Cet algorithme est inspiré principalement du mécanisme de mutation et de sélection de la théorie de l'évolution. Dans sa première version, appliquée au problème aérodynamique [Beyer and Schwefel, 2002], le déroulement de l'optimisation se base principalement sur deux règles :

— La production de nouvelles solutions (individus) à partir de la population de solutions existantes, *via* une fonction de mutation. La mutation consiste à modifier légèrement et de manière aléatoire toutes les variables qui composent les solutions à ce problème de conception.
— Si les nouvelles solutions sont meilleures que les précédentes, alors elles sont conservées, sinon on revient à l'ancienne population.

Cet algorithme a permis de donner des performances d'optimisation meilleures que les algorithmes utilisés auparavant pour ce genre de problème.

Dans les nouvelles versions des algorithmes de stratégie d'évolution [Beyer and Schwefel, 2002], l'étape de génération de nouvelles solutions se rapproche de celles des autres familles d'algorithmes évolutionnaires. La génération de nouvelles solutions est faite par une fonction de mutation et une autre fonction de croisement qui combine des solutions pour en produire de nouvelles.

Algorithmes génétiques : les algorithmes génétiques [Holland, 1973], proposés par Holland en 1973, sont considérés comme étant les algorithmes évolutionnaires les plus populaires et les plus efficaces pour la résolution de problèmes d'optimisation combinatoire. Ces algorithmes génétiques ont été largement appliqués pour des problèmes d'optimisation en recherche et en ingénierie [Ross and Corne, 1994], [Hüe, 1997].

Les algorithmes génétiques (AG) simulent de manière détaillée le processus de l'évolution des espèces, présentée dans la théorie de Charles Darwin. Les AG simulent tous les mécanismes de la théorie de l'évolution [Holland, 1973], [Hüe, 1997] et non pas que le principe global de mutation et remplacement comme c'est le cas des algorithmes de stratégie d'évolution. En effet, dans cette famille d'algorithmes cinq parties essentielles sont à définir en fonction du problème à résoudre ; l'encodage des solutions en chromosomes, la politique de sélection, les fonctions de croisement et de mutation et une fonction objectif (fitness) pour évaluer la qualité des chromosomes (solutions).

Dans la section suivante de ce chapitre (Section 2.3), nous présentons, plus en détail, les algorithmes génétiques en expliquant le principe de chacune de ces cinq parties qui sont à la base de la conception d'un AG pour l'optimisation.

Programmation génétique : la programmation génétique (PG), proposée en 1992 [Koza, 1994], est considérée comme étant une extension des algorithmes génétiques [Binitha and Sathya, 2012]. Nous parlons dans ce cas de programmation et non pas d'algorithme car le but est de trouver le bon programme informatique pour résoudre un problème donné et non pas juste une solution pour un problème d'optimisation comme c'est le cas pour les AG.

En effet, à l'inverse des autres algorithmes évolutionnaires, la solution à encoder en chromosome, est un programme informatique. Cela implique le fait que les solutions, dans la PG, n'ont pas forcément la même taille ni la même structure. Dans le traitement itératif de recherche, de nouvelles solutions (programmes) sont produites en rajoutant ou en enlevant des variables ou des fonctions aux programmes encodés dans les chromosomes [Koza, 1994].

Un algorithme de programmation génétique se déroule comme suit [Binitha and Sathya, 2012] :

1. Génération de la population initiale de solutions. Plusieurs programmes comportant des variables et des fonctions.
2. Évaluation des solutions de la population. Chaque programme est exécuté et sa qualité (fitness) est estimée en fonction de son degré de résolution du problème en question.
3. Création de nouvelles solutions (programmes) :
   (a) Garder les meilleures solutions de la population.
   (b) Générer de nouvelles solutions par mutation et croisement.
4. Revenir à l'étape 2.

2.2.2 Algorithmes à base d'intelligence distribuée

Ces algorithmes, de leur nom en anglais "Swarm Intelligence", sont plus récents que les algorithmes évolutionnaires, [Bonabeau et al., 1999], ils se sont largement développés depuis le début des années 2000. Ces algorithmes sont inspirés principalement du comportement collectif

des animaux, en particulier chez les insectes (voir Figure 2.3). Le comportement collectif, de ces espèces, est mis en place de manière naturelle, dans le but de se défendre contre d'autres espèces ou dans le but de trouver des sources de nourriture [Bonabeau et al., 1999]. Contrairement aux algorithmes évolutionnaires, ces algorithmes ne se basent pas sur l'évolution d'une population de solutions, mais se basent sur un travail collaboratif d'un ensemble d'agents *via* des mécanismes de communication afin de résoudre des problèmes.

Les espèces, telles que les fourmis ou les abeilles, arrivent à survivre et à s'adapter à leur environnement grâce à un comportement social et un travail collectif. Dans ce comportement social collectif, de manière individuelle, chaque agent a un comportement réactif, aléatoire et que nous ne pouvons pas considérer comme intelligent. La somme de ces comportements, individuels réactifs, génère un comportement global qui permet au groupe de s'adapter et de prendre des décisions que nous pouvons qualifier d'intelligentes, et cela *via* des mécanismes de communication non centralisés.

Les algorithmes à base d'intelligence distribuée implémentent cette intelligence collective dans le but de traiter des tâches de résolution de problèmes ou d'optimisation combinatoire. Différents algorithmes bio-inspirés à base d'intelligence distribuée existent dans la littérature [Bonabeau et al., 1999], et la plupart de ces algorithmes sont inspirés des comportements collectifs de groupes d'animaux.

Ces algorithmes d'optimisation inspirés d'intelligence distribuée peuvent être définis sur la base des cinq principes suivants, présentés dans [Binitha and Sathya, 2012] :

1. **La proximité :** la population (groupe d'agents) doit pouvoir effectuer des calculs simples en matière de temps et d'espace.
2. **La qualité :** la population doit être capable de réagir aux facteurs de qualité de son environnement.
3. **La réponse diversifiée :** la population ne doit pas engager de tâches de recherche dans des espaces non productifs.
4. **La stabilité :** la population ne doit pas changer de comportement pour n'importe quel changement dans son environnement.
5. **L'adaptabilité :** la population doit pouvoir changer de comportement quand cela est bénéfique.

Dans ce qui suit nous présentons l'algorithme de colonies de fourmis [Dorigo et al., 1996] et l'algorithme d'optimisation par essaims particulaires [Kennedy and Eberhart, 1995], deux des plus importants algorithmes bio-inspirés à base d'intelligence distribuée.

Optimisation par essaims particulaires : cette approche est une méta-heuristique bio-inspirée proposée par Kennedy and Eberhart en 1995 [Kennedy and Eberhart, 1995], elle est connue sous le nom de PSO (*"Particle Swarm Optimization"*). Cet algorithme d'optimisation s'inspire du comportement social d'un groupe d'oiseaux à la recherche de nourriture.

En pratique, dans un groupe d'agents (oiseaux), chacun représente une solution et se déplace dans l'espace de recherche. Les déplacements des agents sont guidés par un certain nombre de règles qui simulent le mécanisme de communication entre les agents. Dans l'algorithme, chaque

agent est représenté par quatre éléments, sa position actuelle dans l'espace de recherche (solution actuelle), la meilleure solution qu'il a trouvée depuis le début du processus de recherche, la meilleure solution trouvée par son voisinage depuis le début et la vitesse de déplacement de l'agent dans l'espace de recherche. Ces 4 éléments ont une influence sur les déplacements futurs de l'agent. En effet, à chaque itération, un agent ajuste sa vitesse et sa direction de déplacement dans l'espace de recherche en fonction de sa meilleure solution et de la meilleure solution du voisinage.

Les algorithmes d'optimisation par essaims particulaires se déroulent comme suit :

1. L'initialisation du processus de recherche en assignant aléatoirement une position à chaque agent.
2. L'évaluation de la qualité de chaque agent en utilisant la fonction *fitness*.
3. Mettre à jour la valeur de la meilleure solution trouvée pour chaque agent.
4. Mettre à jour la valeur de la meilleure solution du voisinage, celle de l'agent portant la solution la mieux évaluée du groupe.
5. Mettre à jour la position et la vitesse de déplacement de chaque agent.
6. Revenir à l'étape 2.

Algorithmes de colonies de fourmis : ces algorithmes sont inspirés du comportement social et collaboratif d'un groupe de fourmis formant une colonie [Dorigo et al., 1996]. Les fourmis d'une même colonie, dans leur tâche de recherche de sources de nourriture, abordent un comportement collaboratif intelligent. En effet, des fourmis éclaireuses partent de la colonie et se dispersent de manière aléatoire à la recherche de sources de nourriture autour de la colonie. Chaque fourmi qui trouve une source de nourriture dépose des phéromones sur son trajet, de la source de nourriture vers la colonie. Les autres fourmis de la colonie empruntent, de manière instinctive, les chemins vers les sources de nourriture en suivant les phéromones déposées, et déposent à leur tour des phéromones sur le chemin emprunté. De cette manière, plusieurs sources de nourriture peuvent être exploités en même temps. Le chemin vers la source la plus intéressante, ayant le chemin le plus court, aura une plus forte intensité en phéromones vu que les fourmis vont l'emprunter plus souvent, ce qui va permettre à la colonie d'exploiter cette source davantage. Les autres chemins seront de moins en moins empruntés parce que l'intensité des phéromones va baisser par rapport au chemin le plus court. Les fourmis déposent les phéromones de manière instinctive et non pas intelligente. Le fait que les fourmis suivent, de manière instinctive, les phéromones, en fonction de leurs intensités, crée un mécanisme de communication non centralisée produisant un comportement global intelligent efficace.

Le premier algorithme de colonie de fourmis a été proposé en 1999 par Dorigo et Di Caro [Dorigo et al., 1996], dans le but de trouver le plus court chemin dans un graphe en simulant le comportement des fourmis présenté auparavant. L'algorithme initial n'a pas été destiné à la résolution des problèmes d'optimisation, mais avec le temps, cet algorithme s'est développé et plusieurs variantes ont été proposées pour traiter différentes tâches de résolution de problèmes et d'optimisation combinatoire [Dorigo et al., 1996].

Un algorithme de colonie de fourmis, pour l'optimisation, construit de manière incrémentale la solution au problème. Chaque déplacement d'une fourmi correspond à un déplacement vers

une position adjacente dans l'espace de recherche et à l'ajout d'une partie à la solution qui était vide au départ. L'algorithme se base sur deux règles principales, que nous décrivons comme suit :

— **Construction de la solution :** cette fonction exécute le processus de construction de la solution, dans lequel les fourmis artificielles se déplacent à travers les états adjacents de l'espace de recherche en fonction d'une règle de transition. Ces fourmis créent de manière incrémentale les solutions.
— **Mise à jour des phéromones :** cette fonction effectue des mises à jour des traces de phéromones. Les mises à jour sont faites soit après création de solutions complètes, soit à chaque itération sur des solutions partielles. Cette fonction inclut le mécanisme de l'évaporation des traces de phéromones des solutions moins bonnes. L'évaporation guide les fourmis, dans leurs déplacements, afin d'oublier les mauvaises solutions créées au début de la recherche.

2.2.3 Algorithmes écologiques

Les différents écosystèmes qui composent le monde naturel sont à la source de plusieurs algorithmes pour l'optimisation. Parmi ces algorithmes, nous pouvons citer les algorithmes évolutionnaires et ceux à base d'intelligence distribuée, que nous avons présentés auparavant. D'autres écosystèmes et comportements dans la nature peuvent être une source d'inspiration pour d'autres types d'algorithmes, en particulier les interactions entre les organismes vivants dans leur environnement. Ces interactions peuvent être entre des agents de la même espèce ou entre différentes espèces et même entre des agents d'une espèce et son environnement naturel (air, sol, eau, etc.) [Binitha and Sathya, 2012]. Ces comportements d'interactions complexes entre différents organismes ont permis de proposer des algorithmes d'optimisation ayant un fonctionnement plus complexe que ceux des autres catégories.

Dans la nature, les interactions peuvent être coopératives, comme c'est le cas pour les animaux vivant en troupe où les tâches sont subdivisées et réalisées de manière collaborative. Ces interactions peuvent être aussi, compétitives comme c'est le cas pour les organismes qui profitent des autres espèces à l'image des bactéries.

Parmi les algorithmes écologiques, nous pouvons citer PS2O qui a été proposé en 2008 [Chen and Zhu, 2008], et qui est une extension de l'algorithme d'optimisation par essaims particulaires (PSO). Dans cet algorithme, plusieurs espèces sont impliquées dans le processus de recherche de solutions, où un mécanisme de communication (interaction), inter-espèces et intra-espèces, est implémenté pour renforcer le processus de recherche et le guider vers la solution optimale.

Mehrabian et Lucas ont proposé en 2006, un algorithme inspiré du processus écologique de la colonisation et de l'évolution des mauvaises herbes, dans leur environnement [Mehrabian and Lucas, 2006]. Ces mauvaises herbes envahissent des espaces prolifiques que d'autres organismes occupent et entrent en compétition avec eux pour s'installer définitivement par la suite. L'algorithme commence par une population initiale dispersée sur l'espace de recherche. Par la suite, les agents les mieux évalués ont plus de chance de se reproduire pour permettre à la population de se concentrer autour des régions prolifiques. La reproduction consiste à générer de nouveaux agents (solutions) distribués autour des régions prolifiques afin d'envahir (exploiter) ces espaces de recherche.

2.3 Les algorithmes génétiques

Les algorithmes génétiques (AG), comme nous l'avons présenté auparavant (Section 2.2.1), sont des méta-heuristiques bio-inspirées qui font partie de la famille des algorithmes évolutionnistes [Holland, 1973], [Hüe, 1997]. Un algorithme génétique pour l'optimisation simule le processus d'évolution des individus d'une espèce naturelle dans son environnement [Holland, 1973], [Hüe, 1997].

Dans un processus d'évolution naturelle d'une espèce, les individus qui composent une population, sont représentés par leurs informations génétiques [Holland, 1973]. En biologie, chaque individu a un chromosome unique dans lequel le code génétique de cet individu est sauvegardé. Chaque composante de ce code est sauvegardée dans un gène qui est considéré comme l'unité de base d'un chromosome. Donc, une suite de gènes qui composent un chromosome représente un individu unique de la population.

Dans la figure 2.5, nous présentons un exemple d'une portion d'un des chromosomes de l'être humain. Chez l'être humain, il existe 23 paires de chromosomes pour coder toutes les informations génétiques d'une personne unique. Cependant, pour les algorithmes génétiques nous nous limitons à un chromosome par individu car cela est suffisant pour encoder une solution de la majorité des problèmes d'optimisation. Le chromosome que nous présentons dans la figure 2.5 [2] est composé de 2 968 gènes, où chaque gène prend une valeur parmi 4 valeurs possibles (A, T, C, G). La modification d'une des valeurs des gènes d'un chromosome génère une transformation génétique (mutation) qui modifie un des comportements de l'individu et par conséquent sa qualité d'adaptation à son environnement.

FIGURE 2.5 – Exemple de chromosome humain.

Une population d'individus évolue dans le temps vers des populations de mieux en mieux

2. Image / Photo Credit : Darryl Leja, NHGRI

adaptées à l'environnement [Holland, 1973], [Hüe, 1997]. Cette évolution des individus se fait à travers un mécanisme de reproduction (accouplements et mutations) qui provoque des transformations génétiques au niveau des gènes des chromosomes. Ces transformations génétiques ou chromosomiques permettent de générer de nouveaux individus ayant de nouvelles informations chromosomiques. Dans le processus d'évolution, et d'une génération à une autre, un mécanisme de sélection s'occupe de garder les individus ayant des informations chromosomiques qui assurent la survie de la population et qui permettent de produire de meilleurs individus dans le futur. Les individus sélectionnés peuvent se reproduire, au moment où les autres individus ont tendance à disparaître sans pouvoir se reproduire et transmettre leurs informations chromosomiques [Holland, 1973].

L'idée principale de l'adaptation de ce processus d'évolution génétique pour la résolution d'un problème d'optimisation, qu'a proposée Holland [Holland, 1973], consiste à encoder la solution du problème sous forme d'un chromosome et à considérer par la suite une population d'individus. Cette population d'individus qui représente différentes solutions va évoluer dans un traitement itératif vers des populations de solutions optimales, simulant l'évolution génétique naturelle. Cette adaptation en un algorithme génétique, nécessite la modélisation de cinq parties importantes [Holland, 1973], [Hüe, 1997] :

— La représentation des solutions du problème sous forme chromosomique encapsulant les informations génétiques.
— La définition d'une fonction pour la génération d'une population initiale de chromosomes (individus).
— La définition des fonctions d'accouplement (croisement) et de mutation en fonction de la structure des chromosomes.
— La définition d'une fonction *fitness* qui estime la qualité de la solution encodée dans le chromosome.
— La définition des politiques de sélection et de remplacement, qui permettent de trier les chromosomes d'une population pour en garder certains et éliminer d'autres.

La modélisation, de chacune de ces 5 parties, peut être faite selon différentes stratégies [Hüe, 1997]. La stratégie à mettre en place pour chaque partie est choisie soit en fonction du problème à traiter, soit par optimisation empirique. Une fois ces cinq parties définies, nous pouvons parler d'un algorithme génétique adapté. Le processus de recherche d'un algorithme génétique se déroule de manière itérative répétant les mécanismes d'évaluation, sélection, croisement, mutation et remplacement dans l'ordre. Dans la littérature [Holland, 1973], [Hüe, 1997], deux conditions d'arrêt sont utilisées pour stopper ce traitement itératif. La première est la convergence vers une population qui ne peut plus produire de nouvelles solutions meilleures que celles qui existent. La deuxième condition est un nombre d'itérations fixé empiriquement pour éviter de tourner indéfiniment, s'il n'y pas de convergence. Nous synthétisons ce processus génétique dans la figure 2.6.

Dans un algorithme génétique, plusieurs paramètres [Hüe, 1997] doivent être fixés empiriquement, en plus des stratégies à définir, pour les cinq parties précédemment présentées [Holland, 1973]. Comme nous pouvons le voir sur la figure 2.6, la taille de la population (n), le nombre de chromosomes à sélectionner pour la reproduction (k) ainsi que le nombre de chromosomes à produire par les fonctions de croisement (m) et de mutation (l), sont les principaux paramètres à fixer

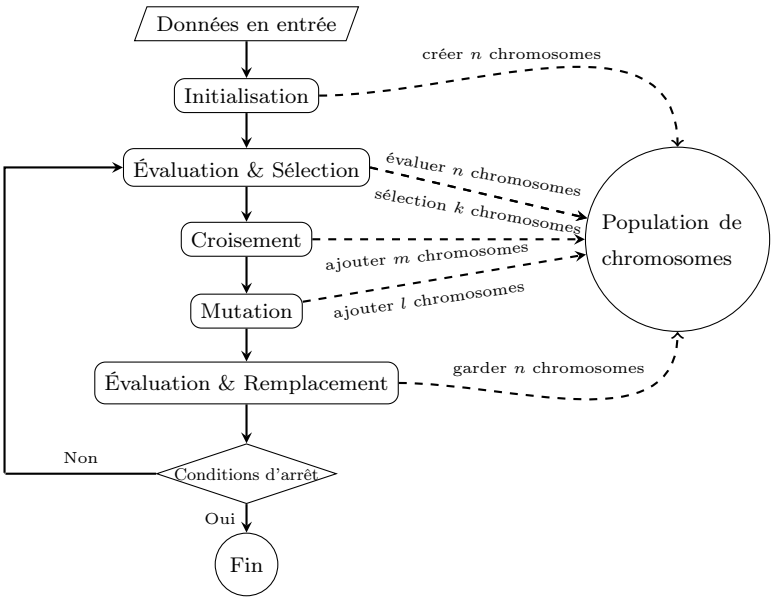


FIGURE 2.6 – Processus itératif d'un Algorithme Génétique (AG).

dans un AG. Ces paramètres ont un impact direct sur le déroulement de l'algorithme en matière d'exploration et d'exploitation de l'espace de recherche [Črepinšek et al., 2013], ainsi que sur la convergence de l'algorithme. En effet, avec une plus large population de chromosomes et des taux élevés de reproduction, l'algorithme va avoir tendance à explorer davantage l'espace de recherche et prendre plus de temps pour converger vers des solutions optimales. Au contraire, une population restreinte avec peu de transformations va avoir tendance à exploiter davantage une région de l'espace de recherche et converger localement. Dans les points suivants de ce chapitre, nous discutons avec plus de détails les différents paramètres de l'algorithme.

Parmi les différentes catégories d'algorithmes bio-inspirés, que nous avons présentées au début de ce chapitre, les algorithmes évolutionnaires ont la particularité de bien pouvoir trouver un équilibre entre une exploration de l'espace de recherche dans sa globalité, et entre une exploitation efficace des régions prometteuses de cet espace [Črepinšek et al., 2013]. Dans le cas des algorithmes génétiques, cet équilibre entre exploitation et exploration est possible grâce aux fonctions de croisement et de mutation et aux différents paramètres de l'algorithme, qui sont ajustés au cours du processus de recherche. Une plus grande diversification et une forte reproduction au début de la recherche, *via* une forte mutation, permettent de produire des chromosomes (solutions) ayant une forte différence génétique, ce qui permet d'explorer au mieux l'espace de recherche. Tandis qu'une faible diversification avec des transformations génétiques locales, *via* des croisements produisant des chromosomes (solutions), qui se rapprochent de ceux qui existent dans la population, permet d'exploiter le potentiel des régions prometteuses de l'espace de recherche.

Grâce à cette efficacité d'exploration et d'exploitation, les algorithmes génétiques sont les méta-heuristiques bio-inspirées les plus utilisées pour l'optimisation combinatoire [Binitha and Sathya, 2012], [Hüe, 1997]. Les AG ont été appliqués à un grand nombre de problèmes, que ce soit dans le cadre des travaux de recherche théorique ou pour la résolution de problèmes d'ingénierie

[Ross and Corne, 1994], [Hüe, 1997].

Dans les sections suivantes, nous présentons chacune des cinq parties essentielles pour l'implémentation d'un algorithme génétique d'optimisation. Nous présentons par la suite, les principales applications des algorithmes génétiques pour la traduction automatique.

2.3.1 Encodage

La représentation génétique d'une solution à un problème d'optimisation est appelée encodage dans le cadre des AG [Holland, 1973], [Hüe, 1997]. L'encodage consiste à trouver la bonne représentation, sous forme chromosomique d'une solution afin de pouvoir lui appliquer, par la suite, des croisements et des mutations et aussi pouvoir définir une fonction de *fitness* afin d'évaluer la qualité de la solution encodée dans le chromosome.

De manière classique la représentation génétique d'une solution se fait sous forme d'une suite de caractères, généralement binaires, produisant un vecteur d'une certaine taille fixe [Hüe, 1997]. Afin de mieux expliquer l'encodage, prenons l'exemple du problème de satisfaisabilité booléenne (SAT) [Zhang et al., 2001] à 9 variables booléennes ($V = x_1, ..., x_9$) et 100 clauses logiques ($C = c_1, ..., c_{100}$). SAT est considéré comme le problème NP-complet de référence [Zhang et al., 2001], dans lequel le but est d'affecter une valeur booléenne à chaque variable afin de satisfaire le maximum de clauses logiques. Pour ce problème, un chromosome est encodé sous forme d'un vecteur de 9 cases (gènes) (Figure 2.7). Chaque case prend une valeur soit de 1 (vrai) soit de 0 (faux).

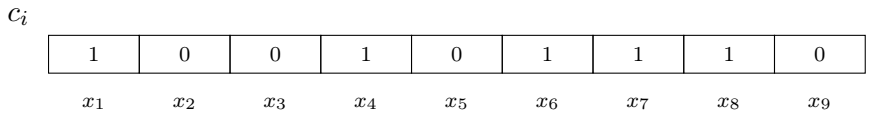


FIGURE 2.7 – Exemple de représentation génétique pour un problème SAT.

L'importance, tout comme la difficulté, de l'encodage apparaît davantage dans le cas des problèmes d'optimisation ayant un espace de recherche à valeurs non numériques. En effet, dans le cadre du traitement automatique de texte, les solutions des problèmes d'optimisation sont souvent des ensembles de mots de différentes tailles et qui peuvent englober d'autres informations en plus des mots. Le problème de traduction automatique au niveau du décodeur [Otto and Riff, 2004] et le problème du résumé automatique [Bossard and Rodrigues, 2015] sont deux exemples d'utilisation des algorithmes génétiques dans le cadre du traitement de texte. Pour le premier, une solution est composée de deux éléments ; une suite de mots représentant une traduction et un vecteur de valeurs numériques représentant les alignements entre les mots de la phrase source et ceux de la traduction. Pour le deuxième problème, une solution est composée d'un ensemble de mots de tailles variables. Le problème de génération des alignements au niveau des mots ou au niveau des segments, à partir de corpus parallèles, est un autre problème d'optimisation pour lequel une solution ne peut être encodée sous forme de vecteur de valeurs numériques. Une solution dans ce cas, est une suite de paires de segments sources et cibles. Dans ce genre de problème, il est souvent difficile de trouver la bonne représentation génétique qui retranscrit idéalement les solutions du problème. Une solution à ce problème de représentation, est la transformation de l'espace de recherche en un espace numérique permettant d'encoder les solutions sous forme de

vecteurs de valeurs numériques.

L'encodage ou la représentation génétique des solutions d'un problème d'optimisation doit permettre aux individus (chromosomes) de se reproduire et cela en ayant une structure facilitant la définition des fonctions de croisement et de mutation.

Dans les sections suivantes, et afin de présenter et illustrer les différentes parties d'un AG, nous nous basons sur le problème SAT à 9 variables et 100 clauses avec l'encodage chromosomique présenté dans la figure 2.7. Dans la Section 2.4, nous présentons des applications des algorithmes génétiques pour la traduction automatique tout en présentant différents exemples d'encodage.

2.3.2 Initialisation

L'initialisation consiste à créer une première population de chromosomes (population initiale). Pour l'initialisation, une stratégie est à définir et un paramètre à fixer. La stratégie est celle de la diversification des chromosomes de la population initiale et le paramètre est celui de la taille de cette population. Ces deux critères ont une forte influence sur le déroulement et l'évolution de l'AG [Hüe, 1997]. En effet, afin d'explorer au mieux l'espace de recherche et exploiter plusieurs régions prometteuses [Črepinšek et al., 2013], à l'initialisation, il faut produire un nombre suffisant de chromosomes dispersés de manière équilibrée sur la globalité de l'espace de recherche. D'un autre côté, ces deux paramètres vont avoir un impact sur la durée de convergence de l'AG. Avec une population large et diversifiée, l'AG aura besoin d'un important nombre d'itérations pour pouvoir converger vers une population optimale. À l'inverse, une population restreinte peut provoquer une convergence plus rapidement vers un optimum qui risque d'être local.

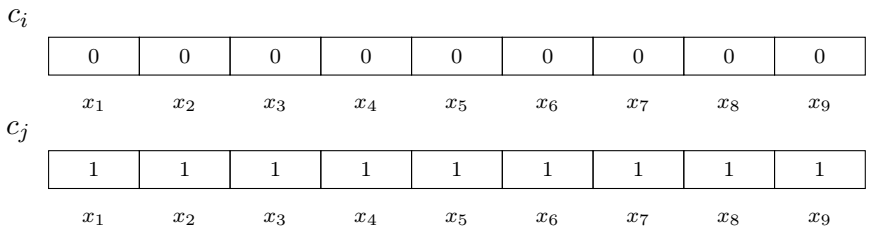


FIGURE 2.8 – Exemple de deux chromosomes dispersés dans l'espace de recherche.

Si nous reprenons l'exemple SAT à 9 variables binaires, l'espace de recherche est composé de $2^9 = 512$ solutions possibles. Dans cet espace, il faut produire, par exemple, une population initiale de 50 chromosomes. Ces 50 chromosomes doivent être répartis sur l'espace des 512 solutions possibles. Dans la figure 2.8, nous présentons deux chromosomes qui encodent deux solutions à ce problème. La distance entre ces deux chromosomes est la plus grande distance que peut avoir deux solutions, dans l'espace de recherche et cela en se basant sur une distance euclidienne entre les deux vecteurs. Ces deux chromosomes vont permettre d'explorer deux régions distantes de l'espace de recherche.

2.3.3 Reproduction

Le croisement et la mutation sont les deux mécanismes de reproduction dans les AG [Holland, 1973], [Hüe, 1997]. Ces deux mécanismes (fonctions) permettent de produire de nouveaux chromosomes

à partir de ceux qui existent dans la population et cela en appliquant des transformations génétiques au niveau des gènes des chromosomes. Dans ce qui suit, nous présentons ces deux fonctions ainsi que leurs différences.

Croisement

Le croisement est la fonction qui simule le mécanisme d'accouplement dans le processus naturel de l'évolution d'une espèce. Pour appliquer le croisement, deux chromosomes de la population sont sélectionnés de manière aléatoire, ou suivant une certaine stratégie de sélection. Ces deux chromosomes sont considérés comme des chromosomes parents c_{p1} et c_{p2}. L'accouplement consiste à croiser les deux chromosomes parents en mixant leurs gènes. Le croisement produit deux chromosomes enfants c_{e1} et c_{e2} qui se partagent les informations génétiques des parents. Le croisement donc, permet de produire deux nouvelles solutions proches des solutions parentes. Idéalement, le croisement permet de produire des chromosomes encodant des solutions qui combinent des parties intéressantes (gènes), qui existaient séparément dans les solutions encodées dans les chromosomes parents.

Il existe plusieurs stratégies pour définir la fonction de croisement [Hüe, 1997]. La stratégie la plus utilisée consiste à choisir aléatoirement un point (position) de croisement (Figure 2.9). Les deux parents sont coupés en deux parties, de part et d'autre du point de croisement, et les chromosomes enfant sont produits en sélectionnant de manière alternative une partie d'un des deux parents comme nous l'illustrons dans la figure 2.9.

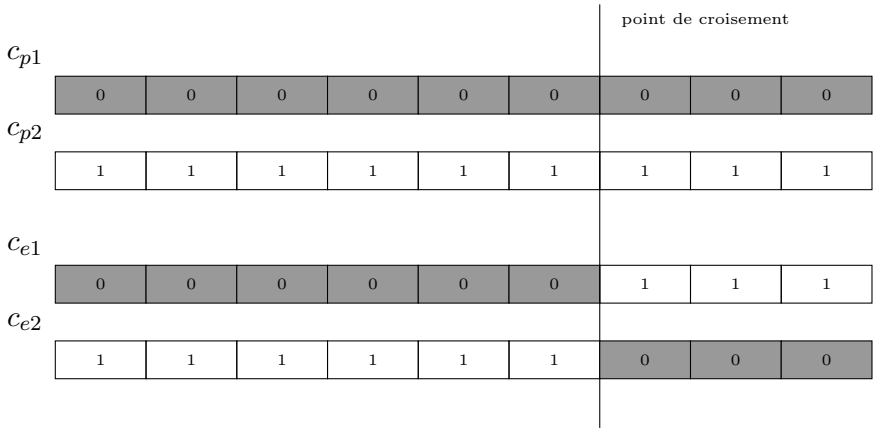


FIGURE 2.9 – Exemple de fonction de croisement à 1 point.

Afin de produire des chromosomes enfants avec une plus forte différence par rapport aux chromosomes parents, plusieurs points de croisement peuvent être sélectionnés. Cette stratégie est appelée dans la littérature "*croisement par N-points*" [Hüe, 1997]. Une autre stratégie est *le croisement uniforme*, dans lequel les chromosomes enfants sélectionnent aléatoirement de quel parent copier chaque gène. Le premier chromosome enfant (c_{e1}) sélectionne aléatoirement à partir de quel parent copier le premier gène (x_1), et le deuxième chromosome enfant (c_{e2}) copie automatiquement ce gène à partir de l'autre parent, ainsi de suite jusqu'au dernier gène.

Les croisements produisent des chromosomes qui se rapprochent de la population actuelle, car ces nouveaux chromosomes reprennent les gènes des chromosomes parents. Ce mécanisme permet

d'exploiter le potentiel de la population actuelle en combinant les différentes caractéristiques génétiques sauvegardées dans les chromosomes parents. D'un autre côté, le nombre de fois que la fonction de croisement est appliquée à chaque itération (génération) influence automatiquement le nombre de nouveaux chromosomes à produire, donc cela influence le degré d'exploitation de la région dans laquelle la population se regroupe. Ce nombre de fois que le croisement est appliqué est un paramètre important à optimiser et à adapter au cours du processus de recherche génétique.

Mutation

La mutation consiste à appliquer une transformation génétique sur un chromosome pour produire un nouveau chromosome différent [Holland, 1973] [Hüe, 1997]. En pratique, un chromosome est sélectionné aléatoirement à partir de la population, et l'un de ses gènes est sélectionné de manière aléatoire également. La modification, de manière aléatoire, de la valeur de ce gène permet de produire un nouveau chromosome.

À l'inverse du croisement, la mutation permet de produire des chromosomes ayant des gènes différents des gènes encodés dans les chromosomes de la population. Cela permet à l'algorithme de se déplacer vers des régions différentes dans l'espace de recherche et de diversifier la population. Cela permet d'explorer davantage l'espace de recherche et éviter la convergence rapide vers des optimums locaux. Pour accélérer ce processus de diversification au cours du déroulement de l'AG, plusieurs mutations peuvent être appliquées sur le même chromosome en modifiant plusieurs gènes de manière aléatoire. Le nombre de chromosomes à muter à chaque itération est un autre paramètre à fixer pour ajuster le processus de diversification.

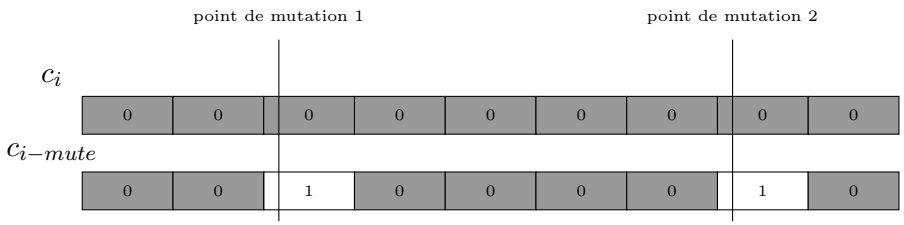


FIGURE 2.10 – Exemple de fonction de mutation.

Dans la figure 2.10, nous prenons un chromosome encodant une solution pour le problème SAT à 9 variables. La mutation appliquée dans l'exemple consiste à sélectionner aléatoirement deux points de mutation (deux gènes) et modifier leurs valeurs. Comme les variables de ce problème sont binaires donc la modification des nouvelles valeurs des gènes consiste à inverser leurs valeurs. Dans d'autres problèmes d'optimisation avec des gènes ayant un plus grand ensemble de valeurs possibles, la mutation consiste à choisir aléatoirement une autre valeur à partir de l'ensemble.

2.3.4 Fonction de fitness

"Fitness" est un terme anglais traduit en français par *aptitude*, qui définit la capacité d'un individu à pouvoir faire une certaine action. Dans le cadre des algorithmes génétiques, ce terme (*"fitness"*), est utilisé pour définir l'aptitude d'un chromosome à résoudre le problème en question. Par analogie à la théorie de l'évolution [Holland, 1973], la *fitness* d'un chromosome est sa

qualité ou son degré d'adaptation à son environnement.

Dans la conception d'un AG, une fonction *fitness* doit être définie afin de pouvoir évaluer la qualité des chromosomes d'une population et de les ordonner par la suite. Toutes les décisions de sélection et de remplacement au cours du déroulement d'un algorithme génétique se basent sur la fonction *fitness*. Par conséquent, la définition d'une bonne fonction d'évaluation est primordiale pour assurer la bonne convergence de l'algorithme vers la solution optimale. Cette fonction est souvent facile à définir car le problème à résoudre est connu à l'avance.

Si nous revenons à l'exemple du problème SAT à 9 variables binaires et 100 clauses logiques à satisfaire. La meilleure solution est celle qui permet de satisfaire la totalité des 100 clauses. D'une manière globale, la solution optimale est celle qui maximise le nombre de clauses satisfaites (Formule 2.1), car la solution qui maximise la totalité des 100 clauses peut ne pas être atteignable si la structure des clauses ne le permet pas.

$$Fitness(c) = \sum_{i=1}^{100} (correcte(clause_i)) \tag{2.1}$$

Où, 100 est le nombre de clauses à satisfaire et *"correcte(clause$_i$)"* est une fonction qui vérifie si une clause *"clause$_i$"* est vraie ou non.

D'autres problèmes d'optimisation sont plus complexes car l'évaluation se base sur plusieurs critères. Une optimisation à multi-critères risque de provoquer des situations de conflit. Un critère décide qu'un chromosome est meilleur que l'autre, et inversement pour un autre critère. Pour ce genre de problème une étude plus approfondie du problème est nécessaire pour pouvoir définir la fonction *fitness* adéquate.

2.3.5 Sélection et remplacement

Dans la théorie de l'évolution naturelle, les deux mécanismes de sélection et de remplacement assurent l'évolution de la population dans le temps vers une population mieux adaptée à son environnement [Holland, 1973]. D'une part, la sélection consiste à choisir, à chaque génération (voir Figure 2.6), quels chromosomes de la population peuvent se reproduire par croisement et mutation. D'autre part, le remplacement est un mécanisme appliqué à la fin de chaque génération (voir Figure 2.6), et il consiste à choisir quels chromosomes garder pour la génération suivante, parmi ceux de la population et parmi les nouveaux chromosomes résultant de la reproduction.

Les décisions prises par les deux mécanismes sont basées sur les *fitness* des chromosomes et les stratégies de sélection et de remplacement. Dans les deux points qui vont suivre, nous expliquons comment sont définis ces deux mécanismes en pratique dans un AG.

Sélection

La politique de sélection des individus pour la reproduction est une étape cruciale dans le déroulement de l'AG, elle a un impact direct sur la convergence de l'algorithme [Holland, 1973], [Črepinšek et al., 2013]. En effet, si la reproduction n'est permise qu'aux chromosomes les mieux

évalués de la population, ayant les meilleures *fitness*, alors l'algorithme aura tendance à converger plus rapidement en exploitant des régions particulières et explorant moins l'espace de recherche. Inversement, si la politique de sélection donne la possibilité à des chromosomes faiblement évalués de se reproduire, l'algorithme prendra plus de temps à converger, tout en permettant d'explorer différentes régions de l'espace. Définir une politique de sélection permettant de trouver un bon équilibre entre exploitation et exploration est nécessaire pour pouvoir converger, tout en évitant les minimums locaux.

Dans la littérature des algorithmes génétiques [Hüe, 1997], il existe plusieurs stratégies pour la définition de la politique de sélection. Ces stratégies sont de deux catégories, soit une sélection proportionnelle à la *fitness* globale de la population, soit une sélection ordinale sans prendre en compte la *fitness* globale de la population.

La sélection ordinale, la plus simple, consiste à sélectionner les k meilleurs chromosomes, selon leur *fitness*, parmi les n chromosomes de la population. Dans le cas d'une sélection ordinale, si nous voulons plus de diversité, les k chromosomes à sélectionner peuvent être répartis entre une majorité de bons chromosomes (forte *fitness*) et une minorité de chromosomes moins bons ou mauvais (faible *fitness*). Cette politique de sélection n'est pas souvent utilisée vu qu'elle conduit rapidement vers des optimums locaux et ne permet pas d'explorer de manière efficace l'espace de recherche.

La sélection par roulette est une politique qui prend en compte la qualité globale de la population et qui est largement utilisée dans les algorithmes génétiques. Cette technique arrive à trouver un bon équilibre entre exploration et exploitation de l'espace de recherche. Pratiquement, tous les chromosomes de la population sont placés sur une table de roulette et chaque chromosome va occuper un espace angulaire avec un angle en fonction de sa *fitness* normalisée par la somme des *fitness* des chromosomes de la population, comme nous l'illustrons sur la figure 2.11. La roulette est lancée k fois, et à chaque lancé, le chromosome sur lequel elle tombe, est sélectionné et retiré de la table. De cette manière, les bons chromosomes auront plus de chances (fortes probabilités) d'être sélectionnés tout en laissant de la chance aux mauvais chromosomes d'être sélectionnés.

Fitness des chromosomes

c_1 : 60
c_2 : 25
c_3 : 10
c_4 : 48
c_5 : 5

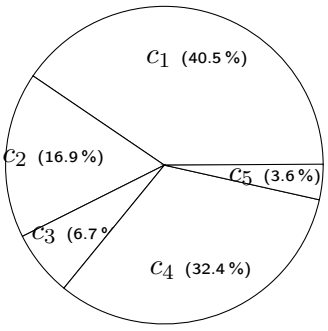


FIGURE 2.11 – Exemple de sélection à base de roulette pour le problème SAT à 9 variables.

Remplacement

Dans chaque itération (génération) de l'algorithme génétique, de nouveaux chromosomes sont ajoutés à la population à travers les fonctions de reproduction. À la fin de chaque génération, $n + l + m$ chromosomes existent dans la population. Un mécanisme doit être défini pour choisir quels chromosomes (n chromosomes) doivent être gardés pour la génération suivante et leur permettre de se reproduire par la suite [Holland, 1973], [Hüe, 1997], [Črepinšek et al., 2013]. Ce mécanisme est appelé remplacement parce que de nouveaux chromosomes peuvent remplacer des chromosomes de la population actuelle et cela selon une stratégie de remplacement à mettre en place.

Dans les AG, la population doit converger vers des populations qui exploitent des régions prometteuses et y trouver la solution optimale au final. La stratégie de remplacement doit permettre aux nouveaux chromosomes de pouvoir se reproduire afin de générer de meilleures solutions. En même temps, cette stratégie doit éliminer les chromosomes faiblement évalués et ne participant pas à la bonne convergence de la population.

Dans la littérature des AG [Holland, 1973], [Hüe, 1997], la stratégie de remplacement la plus intuitive est la sélection des n premiers chromosomes selon leur *fitness*, parmi la population et les chromosomes résultant de la reproduction. Cette stratégie permet de garder les bons chromosomes et leur donne la possibilité de se reproduire dans la génération suivante. Cependant, cette stratégie est brutale et peut mener à une convergence vers un minimum local, si les chromosomes de la population se regroupent dans une même région. Une autre possibilité est de garder l'élite des chromosomes de la population et d'appliquer un remplacement aléatoire sur le reste des chromosomes pour en garder n chromosomes à la fin. La sélection par roulette est utilisée aussi, pour définir la stratégie de remplacement. Le remplacement par roulette donne plus de chances aux bons chromosomes de rester dans la population et en même temps une chance aux autres chromosomes d'être gardés dans la population.

Garder une certaine diversité dans la population est un point très important dans les AG. Cette diversité évite de converger vers des optimums locaux et permet à l'algorithme d'explorer plus de régions de l'espace de recherche. L'existence de chromosomes ayant une faible *fitness* peut être bénéfique dans le processus de recherche génétique car ces chromosomes peuvent encoder des informations génétiques intéressantes (bons gènes). Ces bons gènes peuvent participer à la production de bons chromosomes grâce aux croisements. Un autre point très important, afin d'assurer cette diversité, est l'élimination des doublons au moment du remplacement [Hüe, 1997]. En effet, le processus génétique et les fonctions d'initialisation, de croisement et de mutation, ont tous des comportements aléatoires et non intelligents ce qui peut produire des chromosomes en double. Ces chromosomes en double, dans une même population, peuvent provoquer une convergence rapide vers une région de solution restreinte.

2.4 Application des AG pour la traduction automatique

Les AG ont été utilisés dans un grand nombre de problèmes d'optimisation, que ce soit dans le cadre de travaux de recherche ou dans le cadre d'applications en ingénierie [Hüe, 1997], [Ross and Corne, 1994]. Concernant le traitement automatique des langues (TAL), ces algorithmes ont été appliqués dans le cadre d'un bon nombre de problèmes TAL [Araujo, 2007], [Lempa et al., 2010] à l'image de l'analyse syntaxique et sémantique, l'induction grammaticale,

les résumés et la génération de texte, le clustering de documents et la traduction automatique [Araujo, 2007].

Dans cette partie du manuscrit, nous nous intéressons aux applications des algorithmes génétiques pour la traduction automatique. En effet, les AG ont été appliqués à différentes approches de la traduction automatique [Araujo, 2007], [Nabhan and Rafea, 2005], [Echizen-ya et al., 1996], [Otto and Riff, 2004], [Han, 2001]. La majorité des travaux dédiés à l'application des AG à la TA consiste en l'optimisation des paramètres des algorithmes d'apprentissage ainsi que l'optimisation des paramètres des systèmes de traduction. D'autres travaux proposent des algorithmes génétiques pour la résolution du problème de décodage en TAS. Dans ce qui suit, nous listons les travaux implémentant des AG pour la traduction automatique.

2.4.1 AG pour la TA à base d'exemples de traduction

En 1996, Echizen-ya et al [Echizen-ya et al., 1996] proposent un algorithme génétique pour la production d'exemples de traduction pour un système de TA à base d'exemples de traduction. L'idée de ce travail est l'utilisation des AG pour un apprentissage inductif des exemples de traduction pour la paire anglais-japonais. Un chromosome encode un exemple de traduction composé d'un segment en anglais et sa traduction en japonais. La *fitness* des chromosomes est basée sur un score estimé à partir des retours des utilisateurs. Un utilisateur du système introduit une phrase à traduire et le système lui propose différentes traductions en utilisant la population d'exemples actuelle, et par la suite, l'utilisateur valide les traductions qu'il trouve correctes. De cette manière, la *fitness* d'un chromosome est calculée en fonction du nombre de fois que l'exemple, encodé dans le chromosome, a généré une traduction validée par les utilisateurs par rapport au nombre de fois qu'il a été appliqué. À chaque génération, de nouveaux exemples de traduction sont produits par croisement et mutation. Pour le croisement, deux exemples de traduction ayant des parties communes sont sélectionnés (voir Figure 2.12). Le croisement consiste à croiser les parties différentes entre les deux exemples de traduction pour en produire un nouvel exemple comme illustré dans la figure 2.12.

2.4.2 AG pour l'optimisation des paramètres de GIZA++

En 2005, Nabhan et Rafea [Nabhan and Rafea, 2005] proposent un AG pour optimiser les paramètres de GIZA++ [Och and Ney, 2003] dans le cadre de la traduction automatique statistique. Comme nous l'avons présenté dans le chapitre précédent (voir Section 1.4.2), GIZA++ est un outil de construction de modèles de traduction à base de segments pour la TAS. Les paramètres d'apprentissage de cet outil doivent être ajustés en fonction de la paire de langues et du domaine d'application. Les paramètres de ce système, à valeurs discrètes sont optimisés par une autre méthode, tandis que les paramètres à valeurs réelles sont optimisés par l'AG proposé [Nabhan and Rafea, 2005]. Dans ce dernier, un chromosome encode un vecteur des valeurs des paramètres à optimiser. L'entropie croisée statistique [Nabhan and Rafea, 2005] est utilisée pour définir la fonction *fitness*, pour éviter un jugement subjectif sur les traductions produites. L'entropie croisée est souvent utilisée dans les problèmes d'optimisation en particulier ceux d'alignement de séquences afin d'estimer les probabilités des événements rares. Dans ce cadre, un évènement rare est un alignement rare entre deux segments source et cible. Les chromosomes présentant des scores d'entropie faibles sont sélectionnés pour la reproduction, et de nouveaux chromosomes sont

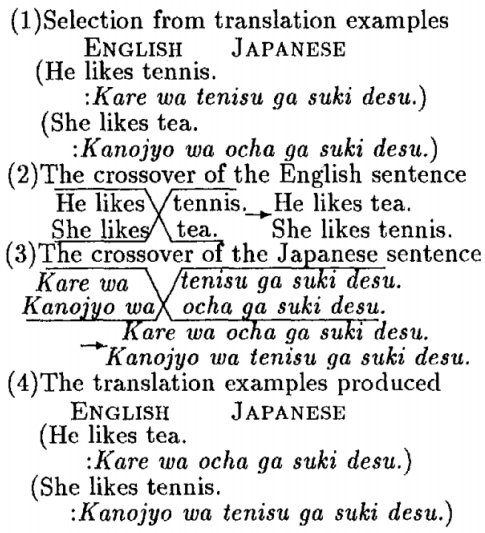


FIGURE 2.12 – Exemple de croisement dans le travail de Echizen-ya [Echizen-ya et al., 1996].

générés avec des opérateurs de croisement et de mutation standards (voir Sections 2.3.3 et 2.3.3).

2.4.3 AG pour le décodage d'un système de TAS à base de mots

En 2004, Otto et Rojas [Otto and Riff, 2004] proposent un algorithme génétique comme décodeur pour la traduction automatique statistique à base de mots. L'algorithme prend en entrée une phrase source et construit à partir de celle-ci la meilleure traduction possible en sortie, celle qui maximise la probabilité de traduction. Les chromosomes de l'AG sont composés de deux structures, la première pour la traduction et la deuxième pour les alignements au niveau des mots comme illustré dans la figure 2.13. La traduction est une suite de mots où la taille des traductions est variable. L'alignement est un vecteur de longueur fixe (taille de la phrase source) où la valeur de chaque case indique la position du mot cible aligné à ce mot source.

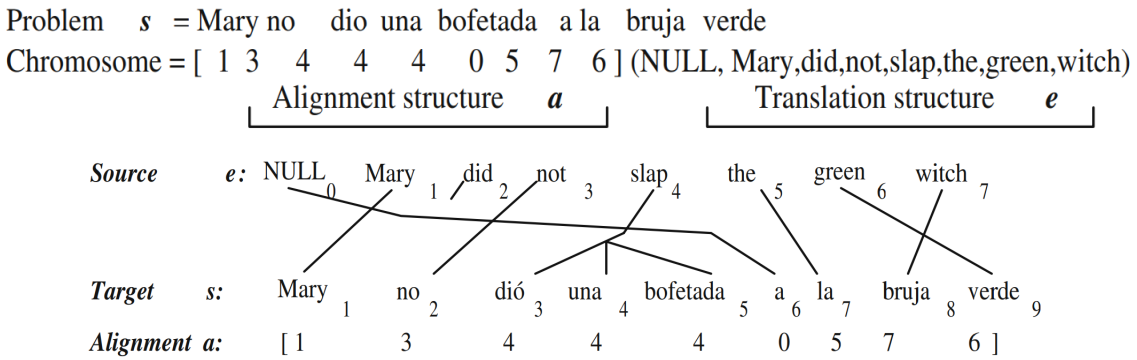


FIGURE 2.13 – Exemple d'encodage de la traduction dans le travail de Otto et Rojas [Otto and Riff, 2004].

Afin d'évaluer les chromosomes, la fonction *fitness* est définie comme étant la probabilité de traduction de la cible encodée dans les chromosomes. Dans ce travail, le modèle IMB4 [Brown et al., 1993] est utilisé. Ce modèle combine différentes probabilités évaluant plusieurs aspects de la traduction : le modèle de traduction à base de mots, la probabilité de fertilité d'un mot source, qui est la probabilité qu'un mot source soit aligné à plusieurs mots dans la cible, la probabilité de distorsion, qui est la probabilité que la position d'un mot dans la source corresponde à la position de sa traduction dans la cible et enfin, la probabilité qu'un mot cible soit aligné au mot NULL. L'approche log-linéaire [Och and Ney, 2003] a été utilisée pour combiner linéairement les quatre probabilités. La fonction *fitness* est définie comme suit :

$$Fitness(c_j) = \sum_{i=1}^{4} log(P_i(s, e^j, a^j)) \tag{2.2}$$

Où c_j est le $j^{éme}$ chromosome de la population. Les P_i sont les 4 scores à combiner. s est la phrase source, e^j et a^j sont respectivement, la traduction et l'alignement encodés dans le chromosome c_j.

La population initiale est générée de manière aléatoire tout en s'assurant d'avoir une certaine diversité de traductions. L'AG utilise différents opérateurs de croisement et de mutation. Pour le croisement, il existe trois fonctions : un croisement à N-points, un croisement d'échange lexical (échange de mots synonymes entre les parents) et un croisement qui sélectionne le meilleur mot traduction à partir des parents selon la probabilité du MT à base de mots. La mutation consiste à appliquer des transformations au niveau des mots, soit en changeant la traduction d'un mot source soit en changeant la position d'un mot cible dans la traduction. Les taux d'application des opérateurs de reproduction sont adaptés au cours du processus d'évolution. En effet, les opérateurs de reproduction (croisement et mutation) qui sont le plus souvent impliqués dans la production de bons chromosomes auront, au cours du processus de recherche, plus de chances d'être appliqués que les autres opérateurs.

Cet algorithme génétique est proposé comme décodeur pour la traduction automatique statistique à base de mots. Les approches statistiques à base de mots ont montré leurs limites par rapport aux approches statistiques à base de segments. Les performances de traduction de ce décodeur génétique sont acceptables mais moins bonnes que celles des décodeurs de référence pour la TAS à base de mots [Otto and Riff, 2004].

2.5 Conclusion et discussion

Dans ce chapitre, nous avons dans un premier temps, présenté de manière globale les algorithmes bio-inspirés pour l'optimisation combinatoire. Par la suite, nous avons présenté de manière plus avancée les algorithmes génétiques en détaillant leurs différentes composantes, les différentes stratégies de conception ainsi que le déroulement du processus d'optimisation génétique. En fin de chapitre, nous avons donné une idée sur les applications possibles des algorithmes génétiques dans le cadre de la traduction automatique.

Une des applications que nous avons présentées consistait en l'implémentation d'un AG comme décodeur pour la traduction automatique statistique [Otto and Riff, 2004]. Ce travail est dédié à la traduction statistique à base de mots, qui n'est plus d'actualité. Les systèmes de

traduction statistiques les plus performants de nos jours sont basés sur des modèles de traduction à base de segments [Koehn, 2004], [Zens et al., 2002]. Cependant, ce travail à l'avantage de proposer un décodeur génétique permettant la manipulation d'une population de traductions complètes tout au long du processus de recherche. D'un autre côté, Langlais [Langlais et al., 2007] a proposé, un décodeur basé sur un algorithme de recherche locale pour la TAS à base de segments (voir Section 1.4.5). Ce décodeur manipule une seule traduction complète et l'améliore au cours du processus de recherche en explorant son voisinage.

Ces deux travaux ont montré l'efficacité des décodeurs manipulant des traductions complètes au cours du processus de recherche, alors que les décodeurs de référence [Koehn, 2004], [Koehn et al., 2007] se basent sur une construction incrémentale des traductions comme nous l'avons présenté dans la Section 1.4.5. Ces systèmes arrivent à proposer des performances de traduction comparables à celles des systèmes de référence. La manipulation de traductions complètes tout au long du processus de recherche minimise le risque de prise de mauvaises décisions. En effet, dans le cas d'un grand espace de recherche le décodeur n'explore pas la totalité de l'espace mais explore les parties prometteuses à travers une méta-heuristique. Le décodeur prend des décisions en fonction de la qualité des traductions manipulées, et la prise de décisions sur la base de traductions partielles augmente le risque de se diriger vers un optimum local.

Au niveau du décodeur des systèmes de TAS à base de segments, il existe deux problèmes d'optimisation combinatoire à résoudre ; le problème de recherche et de construction des traductions (décodage), présenté dans la Section 1.4.4 du Chapitre 1 ; et le problème d'optimisation des poids de la fonction log-linéaire afin de définir une fonction d'évaluation des traductions efficace, présenté dans la Section 1.4.5 du Chapitre 2.

Dans ce travail de thèse, nous proposons une étude de l'application des algorithmes génétiques pour la traduction automatique statistique à base de segments. Le travail consiste en la mise en place d'un système de traduction statistique, à base de segments, dans lequel le décodeur est entièrement basé sur les AG. Deux algorithmes génétiques sont implémentés pour la résolution des deux problèmes d'optimisation majeurs au niveau du décodeur. Dans la littérature ces deux problèmes sont traités par différents algorithmes d'optimisation (voir Section 1.4.4 et Section 1.4.5 du Chapitre 1).

Le choix des algorithmes génétiques se justifie, d'une part par leur efficacité à trouver un bon équilibre entre exploration et exploitation dans de grands espaces de recherche [Črepinšek et al., 2013], comme c'est le cas pour le décodeur des systèmes de TAS. D'une autre part, les AG permettent de manipuler une population de traductions complètes et non partielles tout au long du processus de recherche.

Dans la suite de ce manuscrit de thèse, nous présentons nos propositions d'algorithmes génétiques pour le décodage et l'optimisation des paramètres d'un système de traduction statistique à base de segments. Nous présentons et justifions nos décisions de conception et de choix de stratégies de chaque composante des deux AG proposés.

3

GAMaT : un décodeur à base d'algorithmes génétiques pour la traduction automatique statistique

3.1 Introduction

Comme nous l'avons présenté dans la Section 1.4.4 du Chapitre 1, dans un système de traduction automatique statistique, le décodeur est la composante dans laquelle un algorithme d'optimisation combinatoire est implémenté afin de trouver la traduction optimale $\hat{e}$ pour une phrase source f. Cet algorithme utilise, comme données d'entrée, les modèles statistiques de traduction et de langage, combinés, éventuellement, avec d'autres modèles. Ce problème d'optimisation est défini comme suit :

$$\hat{e} = \underset{e,a}{argmax} \sum_{i=1}^{|h|} \lambda_i \times log(h_i(f, e, a)) \tag{3.1}$$

La traduction optimale $\hat{e}$ doit maximiser la somme log-linéaire pondérée (λ_i) des scores estimés à partir des différents modèles combinés (h_i). Afin de trouver cette traduction optimale, l'algorithme d'optimisation à implémenter doit trouver une segmentation de la phrase source f et une traduction (segment cible) pour chaque segment source, ainsi qu'un alignement (a) entre les segments sources et cibles. Dans la figure 3.1, nous présentons deux solutions possibles (traductions) pour un problème de traduction d'une phrase source *"la maison blanche des USA est différente de celle du Maroc"* du français vers l'anglais.

Dans la figure 3.1, nous pouvons remarquer que pour une même phrase source, il peut y avoir différentes hypothèses de traduction. Afin de différencier ces deux hypothèses, nous calculons un score d'évaluation en utilisant la formule 3.1. À titre d'exemple, nous utilisons les deux scores du modèle de traduction et du modèle de langue avec des poids égaux. Les modèles que nous utilisons dans cet exemple, et dans les autres exemples de ce chapitre, sont issus d'un apprentissage fait sur un corpus parallèle français-anglais que nous présentons dans le chapitre des expérimentations :

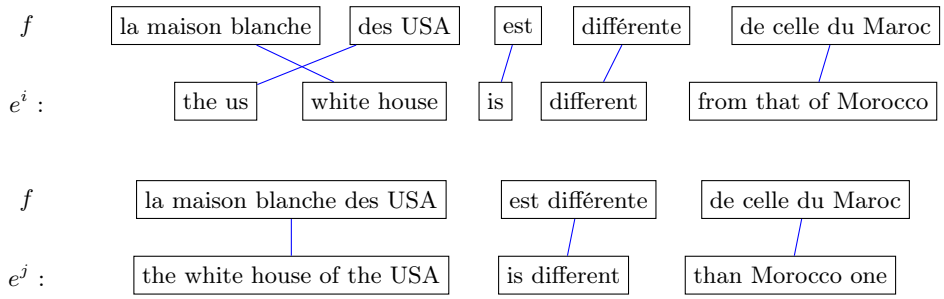


FIGURE 3.1 – Exemples de traductions possibles pour une phrase source en français vers l'anglais.

- e^i="the US white house is different from that of Morocco"
- $P_{ml}(e^i)$ = P_{ml}(the) x P_{ml}(us|the) x P_{ml}(white|the us) x P_{ml}(house|us white) x P_{ml}(is|white house) x P_{ml}(different|house is) x P_{ml}(from|is different) x P_{ml}(that|different from) x P_{ml}(of|from that) x P_{ml}(Morocco|that of)
- $P_{ml}(e^i)$ = 0,5 x 0,15 x 0,09 x 0,25 x 0,015 x 0,08 x 0,48 x 0,315 x 0,12 x 0,09 = $3x10^{-9}$

- $P_{mt}(e^i)$ = P_{mt}(la maison blanche|white house) x P_{mt}(des USA|the us) x P_{mt}(est|is) x P_{mt}(différente|different) x P_{mt}(de celle du Maroc|from that of Morocco)
- $P_{mt}(e^i)$ = 0,6 x 0,15 x 0,95 x 0,8 x 0,35 = 0,024

- score(e^i) = log($3x10^{-9}$) + log(0,024) = -10,143

- e^j="the white house of the USA is different than Morocco one"
- $P_{ml}(e^j)$ = P_{ml}(the) x P_{ml}(white|the) x P_{ml}(house|the white) x P_{ml}(of|white house) x P_{ml}(the|house of) x P_{ml}(USA|of the) x P_{ml}(is|the USA) x P_{ml}(different|USA is) x P_{ml}(than|is different) x P_{ml}(Morocco|different than) x P_{ml}(one|than Morocco)
- $P_{ml}(e^j)$ = 0,5 x 0,3 x 0,25 x 0,15 x 0,09 x 0,45 x 0,31 x 0,48 x 0,35 x 0,05 x 0,1 = $59x10^{-9}$

- $P_{mt}(e^j)$ = P_{mt}(la maison blanche des USA|the white house of the USA) x P_{mt}(est différente|is different) x P_{mt}(de celle du Maroc|than Morocco one)
- $P_{mt}(e^j)$ = 0,28 x 0,8 x 0,19 = 0,042

- score(e^j) = log($59x10^{-9}$) + log(0,042) = -8,6

Nous pouvons voir que la deuxième hypothèse de traduction e^j a un meilleur score (-8,6) que la première hypothèse e^i (-10,143). L'hypothèse e^j est meilleure aussi en termes de scores des deux modèles de traduction et du langage. En plus de la qualité syntaxique de la traduction et la qualité des traductions (paires de segments), nous pouvons remarquer que la taille de la traduction a une influence sur le calcul du score du modèle de langue. En effet, une traduction plus longue a tendance à avoir un faible score du modèle de langue. D'un autre côté, le nombre de segments a une influence sur le calcul du score du modèle de traduction : une hypothèse avec un plus grand nombre de segments a tendance à avoir un faible score du modèle de traduction. En pratique, pour gérer ces deux contraintes, deux scores sont rajoutés dans la combinaison log-linéaire pour normaliser le score final en fonction de la taille de la traduction et le nombre de segments. Pour améliorer la qualité de la fonction d'évaluation des hypothèses de traduction, d'autres scores sont rajoutés à la combinaison. Ces scores évaluent d'autres caractéristiques d'une hypothèse de traduction.

La complexité de ce problème d'optimisation et la taille de l'espace de recherche augmentent rapidement en fonction de la longueur de la phrase source à traduire. En effet, la taille de l'espace de recherche dépend du nombre de segmentations possibles, le nombre de traductions pour

chaque segment et des différentes possibilités de réordonnancement des segments cibles.

Dans la Section 1.4.5 du Chapitre 1, nous avons présenté deux décodeurs différents implémentant deux algorithmes d'optimisation différents. Le premier, celui de MOSES, est fondé sur un algorithme de recherche en faisceau, dans lequel les traductions sont construites de manière incrémentale. Le deuxième est fondé sur un algorithme de recherche locale, dans lequel une traduction complète est traitée tout au long du processus de recherche, en explorant son voisinage. Dans ce chapitre, nous présentons un nouveau décodeur, que nous avons développé, fondé sur les algorithmes génétiques.

Comme nous l'avons présenté dans le chapitre précédent, nous comptons profiter du potentiel des algorithmes génétiques afin de proposer un système de TAS efficace. Les AG nous permettent de manipuler un ensemble d'hypothèses de traduction complètes tout au long du processus de décodage, où d'un côté, le fait de manipuler des hypothèses complètes facilite l'évaluation et améliore les processus de prise de décision. D'un autre côté, la manipulation d'un ensemble de solutions, et non pas qu'une seule, améliore la qualité de l'exploration de l'espace de recherche.

Dans la suite de ce chapitre, nous détaillons l'implémentation de cet algorithme, que nous nommons GAMaT pour *"Genetic Algorithm for Machine Translation"*, en détaillant chaque composante de l'algorithme génétique et les choix des stratégies tout en justifiant chaque décision de conception.

3.2 Encodage

Dans un décodeur d'un système de TAS à base de segments, de manière abstraite, une solution est une traduction complète de la phrase source. Cependant, une solution n'est pas composée que de la traduction mais d'un ensemble d'éléments (voir Figure 3.1), à savoir une suite de segments sources qui représentent la segmentation de la phrase source, une suite de segments cibles qui représentent les traductions des segments sources et un ensemble d'alignements pour sauvegarder les paires de segments sources et cibles. La concaténation des segments cibles produit une hypothèse de traduction complète.

Comme il a été présenté dans le chapitre précédent, dans un algorithme génétique, une solution d'un problème d'optimisation est représentée sous une forme chromosomique. En effet, la solution doit être bien analysée, tout comme le problème traité, afin de pouvoir encoder toutes ses composantes dans un chromosome. Un bon encodage des solutions, facilite l'application des différentes fonctions génétiques et garantit la cohérence des solutions produites par l'algorithme génétique.

Afin de représenter une solution en tant que chromosome pour notre algorithme génétique, nous encodons plusieurs éléments de la solution que nous listons comme suit, et que nous illustrons dans l'exemple de la figure 3.2 :

— **phrase source ($f = \{f_1, ..., f_n\}$) :** un vecteur de mots, de taille fixe n, composant la phrase source. Dans l'exemple, la phrase source f={ *"la", "maison", "blanche", "des", "USA", "est", "différente", "de", "celle", "du", "Maroc"*} est un vecteur de 11 mots français.

— **hypothèse de traduction** ($e = \{e_1, ..., e_m\}$) **:** un vecteur de mots, de taille variable m, composant l'hypothèse de traduction. Dans l'exemple, l'hypothèse de traduction e={ *"the", "us", "white", "house", "is", "different", "from", "that", "of", "Morocco"*} est un vecteur de 10 mots anglais.

— **segmentation de la source** ($\overline{f} = \{\overline{f}_1, ..., \overline{f}_k\}$) **:** un vecteur de taille variable k ($k <= n$). Dans chaque case du vecteur, sont sauvegardés les indices des premier et dernier mots d'un segment source. L'indice de chaque case représente le numéro du segment en question. Dans l'exemple de chromosome de la figure 3.2, nous avons 5 segments sources que nous illustrons sur la phrase source f. Les indices de début et de fin de ces segments sont sauvegardés dans $\overline{f}$={$1:3,\ 4:5,\ 6:6,\ 7:7,\ 8:11$}, où pour le deuxième segment $\overline{f}_2$=4 :5, le premier mot source est à la quatrième position dans f et le dernier mot est à la cinquième position de f.

— **segmentation de la cible** ($\overline{e} = \{\overline{e}_1, ..., \overline{e}_k\}$) **:** un vecteur ayant la même taille k que celle de $\overline{f}$. Dans chaque case du vecteur, sont sauvegardés les indices des premier et dernier mots d'un segment cible. Dans l'exemple de chromosome, nous avons également 5 segments cibles, traductions des segments sources, que nous illustrons sur l'hypothèse e. Les indices de début et de fin de ces segments sont sauvegardés dans $\overline{e}$={$1:2,\ 3:4,\ 5:5,\ 6:6,\ 7:11$}.

— **alignement** ($a = \{a_1, ..., a_k\}$) **:** un vecteur de valeurs entières, ayant la même taille k que celle de $\overline{f}$. La valeur de chaque case a_i représente l'indice (position) du segment cible qui est aligné au segment source $\overline{f}_i$. À titre d'exemple, Si le premier segment de la source $\overline{f}_1$=1 :3 est aligné au deuxième segment de la cible $\overline{e}_2$=3 :4, alors au niveau de l'alignement nous aurons $a_1 = 2$. Dans le chromosome de la figure 3.2 l'alignement a={$2, 1, 3, 4, 5$} est à lire comme suit : a_1=2, a_2=1, a_3=3, a_4=4 et a_5=5 pour dire que le 1^{er} segment source est aligné au $2^{ème}$ segment cible et ainsi de suite jusqu'au $5^{ème}$ segment source qui est aligné au $5^{ème}$ segment cible.

Dans cette représentation génétique, une paire de segments source et cible représente l'unité de base d'une solution (le gène), car nous travaillons dans le cadre de la TAS à base de segments. Nous avons opté pour cette représentation afin de pouvoir définir des fonctions d'initialisation et de reproduction ayant comme unité de base de traitement la paire de segments et aussi, afin de garder une cohérence des solutions. En effet, les segmentations de la source $\overline{f}$ et de la cible $\overline{e}$, avec leurs alignements a permettent de garder la cohérence de la solution et en plus de pouvoir calculer les scores des modèles de traduction et le score de réordonnancement, que nous présentons par la suite. L'hypothèse de traduction complète e, nous permet de calculer le score du modèle de langue. La taille de l'hypothèse de traduction, encodée dans le chromosome, est calculée à partir du vecteur cible e. Quant au nombre de paires de segments, il est calculé à partir d'un des trois vecteurs de segmentation ou d'alignement. Le réordonnancement des segments cibles est sauvegardé dans le vecteur d'alignement a. En effet, les segments sources sont sauvegardés dans l'ordre (voir Figure 3.2), par contre, l'ordre des segments cibles n'est pas forcément monotone par rapport à ceux de la source. Comme nous pouvons le voir dans l'exemple précédent, les positions des deux premiers segments cibles *"white house"* et *"the us"* sont inversées par rapport à l'ordre monotone de leur segments sources alignés *"la maison blanche"* et *"des USA"*.

Dans la figure 3.2, nous présentons un exemple de chromosome pour une phrase source en français traduite vers l'anglais.

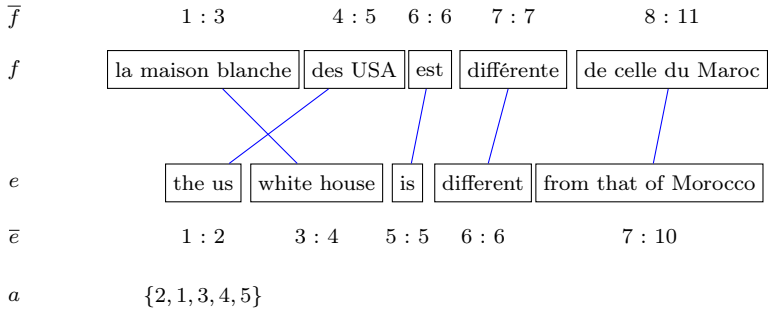


FIGURE 3.2 – Exemple d'un chromosome dans GAMaT.

Dans cet exemple, nous pouvons voir que le réordonnancement entre les deux premiers segments cibles, traductions des deux premiers segments sources, est sauvegardé dans l'alignement $a = \{2, 1, 3, 4, 5\}$, où le premier segment source $\overline{f}_1$ (*"la maison blanche"*) est aligné au deuxième segment cible $\overline{e}_{a_1}$ (*"white house"*), avec $a_1 = 2$. Quant au deuxième segment source $\overline{f}_2$ (*"des USA"*), il est aligné au premier segment cible $\overline{e}_{a_2}$ (*"the us"*), avec $a_2 = 1$.

Toutes les paires de segments sources et cibles que nous utilisons pour la production des chromosomes sont sauvegardées dans la TT. Afin d'illustrer le processus d'initialisation dans GAMaT, nous présentons dans la Table 3.1, une portion d'une TT français-anglais, pour la phrase source f=*"la maison blanche des USA est différente de celle du Maroc"*. Dans cet exemple de TT, nous listons différents segments possibles de la phrase source f et des exemples de traduction pour chaque segment source. Chaque ligne de la TT est composée d'une paire de segments source et cible ainsi que de la probabilité de traduction de cette paire. Dans les TT produites par GIZA++, en plus de la probabilité de traduction, d'autres probabilités et informations qui estiment d'autres aspects de chaque paire de segments y sont sauvegardées.

L'encodage que nous présentons dans GAMaT ne permet pas d'avoir une population de chromosomes ayant la même taille, ni en matière de nombre de mots de l'hypothèse ni en matière de nombre de paires de segments (gènes). Le seul point commun entre les différents chromosomes est la phrase source f. Cet encodage, nous oblige à redéfinir les mécanismes d'initialisation et de reproduction afin d'adapter l'algorithme génétique au problème de décodage et d'assurer une cohérence dans le processus génétique.

3.3 Initialisation

Ayant une représentation chromosomique d'une solution, nous pouvons présenter le processus d'initialisation que nous avons implémenté dans GAMaT. Comme nous l'avons expliqué dans le Chapitre 2, à l'initialisation nous devons produire une population initiale de chromosomes (solutions) suffisamment diversifiée et large afin de pouvoir explorer efficacement l'espace de recherche.

En TAS à base de segments, l'espace de recherche change en fonction de la phrase source à traduire f. Quant à la taille de cet espace de recherche, elle varie en fonction de la taille de f, du nombre de segmentations possibles, du nombre de traductions des différents segments sources ainsi que du fait de la combinatoire dû au réordonnancement des segments de la cible. La totalité des paires de segments avec leur probabilités de traduction sont sauvegardées dans la table de

Source	Cible	Probabilité : P(cible|source)
la	the	0.82164
la	a	0.018540
la	there	0.0019634
la	a that	0.0027157
la	, the	0.0000109
la maison	the home	0.128713
la maison	the house	0.272277
la maison	the museum	0.009009
la maison	house	0.0034653
la maison	my house	0.0049504
la maison blanche	the white house	0.689665
la maison blanche	s white house	0.034556
la maison blanche	the white home is	0.034556
la maison blanche	in the white house	0.034556
la maison blanche des	the white house of	0.569665
la maison blanche des	the white house	0.084556
maison	house	0.517065
maison	home	0.196251
maison	houses	0.0170648
maison	a house	0.0034129
maison	bulding	0.00170648
maison	the home	0.00170648
maison	chamber	0.0068259
maison blanche	white house	0.625
maison blanche	white home	0.25
maison blanche	free house	0.025
maison blanche des	white house of	0.5267
maison blanche des	white home of	0.1523
maison blanche des USA	white house of US	0.3267
maison blanche des USA	US white house	0.6067
maison blanche des USA est	US white house is	0.4267
maison blanche des USA est	USA white house is	0.2287
blanche	white	0.894
blanche	clear	0.01324
blanche	green	0.01423
blanche des	white of	0.734
blanche des	clear of	0.0233
blanche des USA	white of USA	0.234
blanche des USA	white of US	0.1334
blanche des USA est	white of USA is	0.634
des	of	0.7659
des	a	0.0321
des	an	0.0234
des USA	of USA	0.6239
des USA	of US	0.3639
des USA est	of USA is	0.524
des USA est	of US is	0.124
des USA est différente	of USA is different	0.65452
des USA est différente	of US is different	0.35452
USA	USA	0.7239
USA	US	0.1320
USA	united states of america	0.023
USA est	USA is	0.5302
USA is	US is	0.3202
USA est différente	USA is different	0.4022
USA est différente	US is different	0.2568
est	is	0.4430
est	are	0.09230
est	east	0.3204
est	norht east	0.00234
est différente	is different	0.8342
est différente	are different	0.1342
est différente de	is different of	0.7342
est différente de	are different of	0.0954
différente	different	0.8142
différente	distinct	0.1234
différente	not same	0.0032
différente de	different from	0.543
différente de	distinct from	0.1023
différente de	different of	0.343
de	of	0.7453
de	to	0.2345
de	from	0.10453
de	at	0.00234
de celle	of which	0.3232
de celle	from that	0.2122
de celle	of than	0.0022
de celle	from than	0.0122
de celle du	from that of	0.5232
de celle du	from than of	0.0232
de celle du Maroc	from that of Morocco	0.6354
de celle du Maroc	from moroccan one	0.3354
celle	that	0.6542
celle	she	0.1435
celle	her	0.0224
celle du	that of	0.542
celle du	that from	0.0242
celle du Maroc	that of Morocco	0.734
celle du Maroc	moroccan one	0.234
du	of	0.9634
du	due	0.00423
du	a	0.002634
du Maroc	of Morocco	0.87634
Maroc	Morocco	0.92325
Maroc	moroccan	0.020325

TABLE 3.1 – Portion d'une TT français-anglais, pour une phrase source en français.

traduction (TT) que nous avons présentée dans la Section 1.4.2 du Chapitre 1.

Dans GAMaT, 5 fonctions d'initialisation sont utilisées pour créer la première population de chromosomes. La différence entre les 5 fonctions réside dans la stratégie de segmentation de la phrase source et la stratégie de génération des segments cibles (traductions des segments sources). Parmi ces fonctions, deux ont un comportement totalement aléatoire générant aléatoirement une segmentation de la phrase source et une traduction pour chaque segment. Ces 2 fonctions permettent de diversifier la population en produisant des chromosomes encodant des gènes (paires de segments) de différentes qualités, et aussi pour respecter l'aspect aléatoire des algorithmes génétiques, qui doit explorer aléatoirement l'espace de recherche. Les trois autres fonctions ont un comportement guidé afin d'améliorer les processus de segmentation et de traduction. Ces trois fonctions produisent des chromosomes qui encodent des solutions de bonne qualité selon les probabilités du modèle de traduction et cela en favorisant les longs segments, donc un faible nombre de segments.

La structure de l'espace de recherche et celle des chromosomes nous contraint à proposer ces stratégies d'initialisation, qui différent de celles de la littérature des algorithmes génétiques présentés dans la Section 2.3.2 du Chapitre 2. En effet, pour ce problème de traduction, nous avons deux informations qui donnent une idée, *a priori*, sur la qualité de la solution. La première information est que chaque paire de segments a une probabilité de traduction qui estime sa qualité, donc un chromosome constitué de paires de segments ayant de fortes probabilité de traduction a de fortes chances de proposer une bonne traduction. La deuxième information est le nombre de segments du chromosome, où les longs segments ont tendance à mieux traiter les relations linguistiques (réordonnancements, alignements n-n) entre deux langues. Nous nous basons sur ces informations pour varier la qualité des chromosomes de la population en proposant des fonctions produisant des chromosomes encodant des traductions de différentes qualités. Ce genre d'information n'est pas toujours disponible, dans les problèmes d'optimisation classiques, où le processus d'initialisation n'est que production de chromosomes de manière totalement aléatoire.

Avant l'application de n'importe quelle fonction d'initialisation, nous listons tous les segments possibles de la phrase source en utilisant la TT et cela afin de faciliter le processus de segmentation par la suite. Nous sauvegardons les tailles des segments de f dans le vecteur $PH =< ph_1, ..., ph_n >$, où n est la taille de f et chaque case ph_i représente la taille du plus long segment de f commençant par le $i^{ème}$ mot de f et présent dans TT. En effet, dans une table de traduction à base de segments, si un segment de taille k qui commence par le mot w existe dans la TT, alors tous ses sous-segments (préfixes) qui commencent aussi par w, existent dans la TT. Dans le tableau suivant, nous illustrons la mise en place de ce vecteur PH pour l'exemple de la phrase source de la figure 3.2 en se basant sur la TT de la Table 3.1. Dans le Tableau 3.2, le plus long segment commençant par le $2^{ème}$ mot *"maison"* est de taille 5 (*"maison blanche des USA est"*). Les sous-segments de ce dernier *"maison blanche des USA", "maison blanche des", "maison blanche" et "maison"* existent aussi dans la TT et peuvent être utilisés pour la segmentation.

la	maison	blanche	des	USA	est	différente	de	celle	du	Maroc
4	5	4	4	3	3	2	4	3	2	1

TABLE 3.2 – Vecteur PH des tailles des segments d'une phrase source f.

Dans les points qui vont suivre, nous présentons en détail le processus de chaque stratégie de segmentation tout en illustrant leurs comportements à travers la phrase source f= *"la maison blanche des USA est différente de celle du Maroc"* et la portion de TT de la Table 3.1.

3.3.1 Segmentation basée sur les segments les plus longs

La stratégie de segmentation de cette fonction favorise les plus longs segments de la phrase source. Les longs segments ont tendance à produire de bonnes traductions car ils couvrent plus de relations syntaxiques et sémantiques entre la langue source et la langue cible, comparés aux segments courts. Un long segment peut couvrir plus d'alignements de type *n-m* ainsi que des réordonnancements dans la cible.

Algorithm 1 Segmentation basée sur les segments les plus longs : SegmentationGlobale

Variables Globales : c=<f,$e = null$,$\overline{f} = null$,$\overline{e} = null$,a=$null$>, $cpt = 0$, $nbp = 1$
Entrée: $début = 1$, $fin = |f|$, TT
pour $i = 1$ **à** $i = |f|$ **faire**
  $PH_i \leftarrow \max(|s| \ / \ s \in TT \ and \ s = f_i^{|s|+i-1})$
fin pour
si $cpt = |f|$ **alors**
  ReconstructionMonotone($c.\overline{f}, c.\overline{e}, c.f, c.e$)
  Retourner(c)
fin si
si $début \leq fin$ **alors**
  $c.\overline{f}_{nbp} \leftarrow$ SegmentLePlusLong($début$, fin, $c.f$, PH)
  $c.\overline{e}_{nbp} \leftarrow$ ChoixTraduction($c.\overline{f}_{nbp}$, TT)
  $i \leftarrow$ PositionDébutSegment($c.\overline{f}_{nbp}$, $c.f$)
  $j \leftarrow$ PositionFinSegment($c.\overline{f}_{nbp}$, $c.f$)
  $c.a_{nbp} \leftarrow nbp$
  $cpt \leftarrow cpt + |c.\overline{f}_{nbp}|$
  $nbp \leftarrow nbp + 1$
  SegmentationGlobale($début$, $i - 1$, TT)
  SegmentationGlobale($j + 1$, fin, TT)
fin si

En pratique, le processus de segmentation commence par générer le segment le plus long de la phrase source et qui existe dans la TT. Ce premier segment est traduit en sélectionnant sa meilleure traduction (la plus probable) à partir de la TT, cela produit la première paire de segments. Par la suite, le processus de segmentation continue de manière récursive sur les parties à gauche et à droite du premier segment, comme illustré dans l'Algorithme 1. Le processus récursif de cette fonction continue jusqu'à ce que la totalité de la phrase source soit segmentée. Le processus récursif de segmentation génère des segments sources qui ne sont pas dans le bon ordre, donc en sortant du processus récursif nous appliquons un traitement de réorganisation pour remettre les segments sources et cibles dans le bon ordre. Ce traitement de réorganisation est fait *via* la fonction $ReconstructionMonotone(c.\overline{f}, c.\overline{e}, c.f, c.e)$, que nous détaillons dans ce qui suit.

Dans l'Algorithme 1, nous utilisons plusieurs fonctions pour traiter certains aspects de cette procédure d'initialisation. Nous présentons dans ce qui suit ces différentes fonctions :

— **ChoixTraduction :** cette fonction prend un segment source $c.\overline{f}_i$ et lui génère un segment cible $c.\overline{e}_{a_i}$. Le segment cible sélectionné a la probabilité de traduction la plus élevée dans la TT.

— **SegmentLePlusLong :** cette fonction prend une portion de la phrase source (indices de début et de fin) et sélectionne le segment le plus long de la source qui existe dans cette portion.
— **PositionDébutSegment :** cette fonction retourne la position, dans la phrase source $c.f$, du premier mot d'un segment source.
— **PositionFinSegment :** cette fonction retourne la position, dans la phrase source $c.f$, du dernier mot d'un segment source.
— **ReconstructionMonotone :** cette fonction prend les segments sources $c.\overline{f}$ et cibles $c.\overline{e}$, générés par la procédure d'initialisation, et elle reconstruit le chromosome c avec un alignement monotone entre segments sources et cibles. Le fonctionnement récursif de cette première procédure d'initialisation génère des segments de manière non ordonnée, sur différentes portions de la source, donc nous sommes obligés d'appliquer un post-traitement sur les paires de segments pour produire un alignement monotone correct.

Avec cette fonction, nous produisons un seul chromosome ayant la meilleure segmentation possible sur la base de l'heuristique que nous avons définie, à savoir la sélection à chaque étape du segment le plus long. Si, à une étape du processus récursif, nous trouvons plusieurs longs segments avec la même taille, alors le segment ayant la meilleure traduction (probabilité de traduction) est sélectionné. L'alignement entre segments sources et cibles de ce chromosome est monotone, tout comme les autres chromosomes que nous produisons à l'initialisation. Dans la figure suivante 3.3, nous illustrons le résultat de cette fonction d'initialisation.

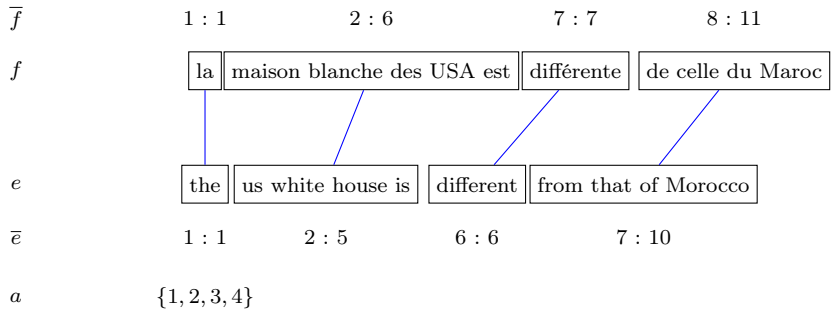


FIGURE 3.3 – Exemple de chromosome résultant de la fonction d'initialisation avec segmentation basée sur les segments les plus longs.

3.3.2 Segmentation de la gauche vers la droite basée sur les segments les plus longs

Pour cette fonction, la phrase source f est segmentée de la gauche vers la droite. Le premier segment est le plus long préfixe (segment à partir de la gauche) de f présent dans TT. La partie de f, non segmentée encore, est traitée de la même manière (voir Algorithme 2), où à chaque fois le plus long segment préfixe est sélectionné est ajouté à la segmentation de la source $\overline{f}$, jusqu'à ce que f soit entièrement segmentée. Pour produire l'hypothèse de traduction, nous sélectionnons, pour chaque segment source, à partir de la TT la traduction ayant la probabilité la plus élevée. En fin de traitement, nous concaténons les segments cibles pour obtenir la traduction e de ce chromosome. Dans la figure 3.4, nous illustrons le résultat de cette fonction appliquée à l'exemple précédent.

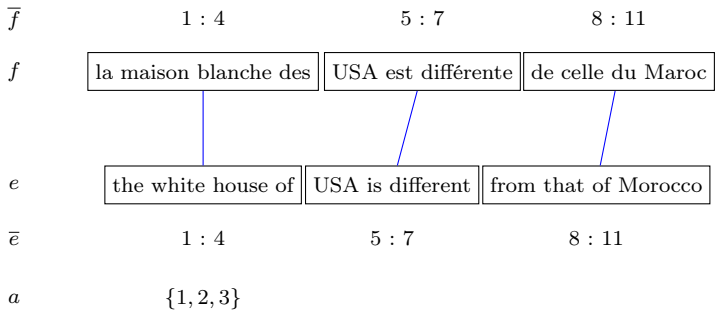


FIGURE 3.4 – Exemple de chromosome résultant de la fonction d'initialisation avec segmentation de la gauche vers la droite basée sur les segments les plus longs.

Algorithm 2 Segmentation de la gauche vers la droite basée sur les segments les plus longs

Entrée: c=<f,e,$\overline{f}$,$\overline{e}$,a>, TT
  pour $i = 1$ **à** $i = |f|$ **faire**
    $PH_i \leftarrow \max(|s| \ / \ s \in TT \ and \ s = f_i^{|s|+i-1})$
  fin pour
  $c.e \leftarrow null$
  $c.\overline{f} \leftarrow null$
  $c.\overline{e} \leftarrow null$
  $c.a \leftarrow null$
  $nbp \leftarrow 1$
  $cpt \leftarrow 1$
  tant que $cpt \neq |f|$ **faire**
    $c.\overline{f}_{nbp} \leftarrow f_{cpt}^{(PH_{cpt}-1)}$
    $c.\overline{e}_{nbp} \leftarrow ChoixTraduction(c.\overline{f}_{nbp}, TT)$
    $c.a_{nbp} \leftarrow nbp$
    pour $\forall w \in c.\overline{e}_{nbp}$ **faire**
      $c.e \leftarrow c.e + w$
    fin pour
    $cpt \leftarrow cpt + PH_{cpt}$
    $nbp \leftarrow nbp + 1$
  fin tant que
  Retourner(c)

Avec cette fonction nous produisons un seul chromosome, car il ne peut y avoir qu'une seule segmentation de la gauche vers la droite basée sur les plus longs segments. Une segmentation de la gauche vers la droite à de fortes chances de produire une bonne segmentation pour les langues qui s'écrivent de la gauche vers la droite comme pour le français ou l'anglais, si nous supposons que les longs segments produisent de meilleures traductions.

3.3.3 Segmentation de la droite vers la gauche basée sur les segments les plus longs

La fonction précédente favorise les longs segments qui se trouvent sur la gauche de la phrase source. Elle ne prend pas en compte les longs segments qui peuvent exister à la droite. Pour bénéficier du potentiel de traduction de ces segments, à la fin de la phrase, nous proposons cette nouvelle segmentation qui se fait de la droite vers la gauche.

Le premier segment est le plus long suffixe de f et qui existe dans TT. Le même processus est appliqué jusqu'à ce que f soit entièrement segmentée. La traduction e est générée de la même manière que pour les deux autres fonctions en sélectionnant la meilleure traduction pour chaque

segment source. À l'inverse des deux fonctions précédentes, dans le vecteur PH, nous sauvegardons les tailles des plus longs segments à droite de la source. la figure 3.5 montre le résultat de cette initialisation pour le même exemple.

Comme pour la fonction précédente, avec cette fonction nous produisons un seul chromosome, car il ne peut y avoir qu'une seule segmentation de la droite vers la gauche basée sur les plus longs segments.

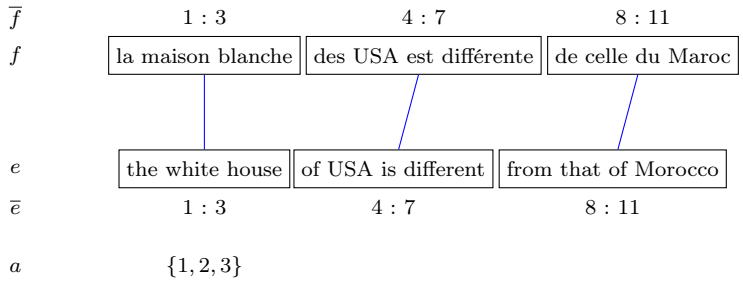


FIGURE 3.5 – Exemple de chromosome résultant de la fonction d'initialisation avec segmentation de la droite vers la gauche basée sur les segments les plus longs.

3.3.4 Segmentation aléatoire de la gauche vers la droite

Pour cette fonction, les processus de segmentation et de génération des segments cibles sont aléatoires, à l'inverse des trois premières fonctions. La phrase source f est segmentée de la gauche vers la droite en sélectionnant à chaque fois au hasard un segment préfixe de la partie non encore segmentée de f. Les traductions des segments sources sont sélectionnées de manière aléatoire à partir de la TT. Dans la figure 3.6, nous présentons un exemple de chromosome résultant de cette segmentation aléatoire et de traduction aléatoire des segments sources.

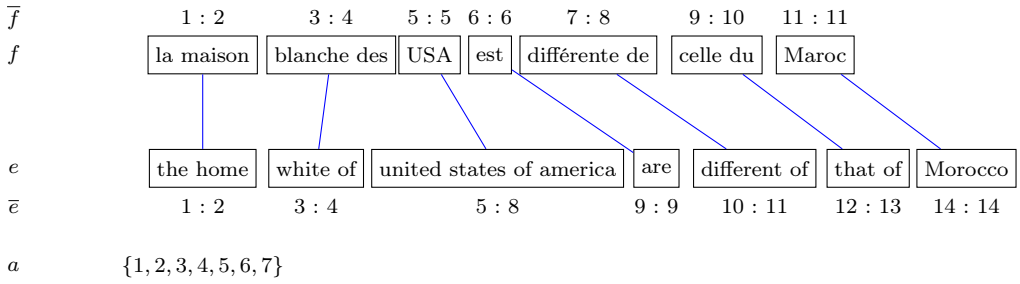


FIGURE 3.6 – Exemple de chromosome résultant de la fonction d'initialisation avec segmentation aléatoire de la gauche vers la droite.

La segmentation et la production des segments cibles sont aléatoires dans cette fonction, donc nous l'utilisons pour produire plusieurs chromosomes afin de diversifier la population initiale.

3.3.5 Segmentation aléatoire de la droite vers la gauche

Dans cette seconde fonction d'initialisation aléatoire, la segmentation se fait de la droite vers la gauche, contrairement à la précédente. Cette fonction, tout comme la précédente, est proposée pour diversifier la population des chromosomes. Le fait de commencer la segmentation aléatoire de la gauche réduit le nombre de segments possible pour les mots se trouvant à la fin de la phrase source. La segmentation aléatoire de la droite vers la gauche redonne à ces mots plus de chances d'appartenir à différents segments.

De manière itérative, nous choisissons un suffixe de manière aléatoire de la partie non encore segmentée de f. Comme pour la fonction précédente, nous utilisons cette fonction pour générer plusieurs chromosomes différents.

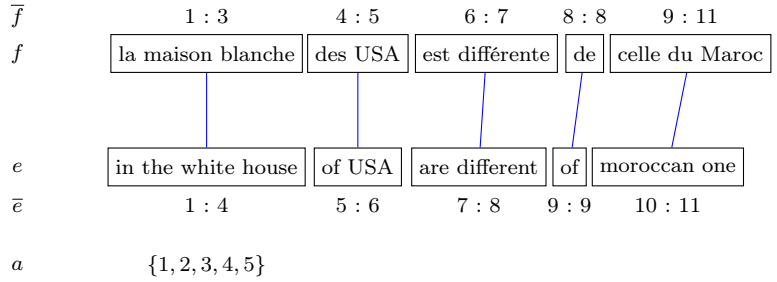


FIGURE 3.7 – Exemple de chromosome résultant de la fonction d'initialisation avec segmentation aléatoire de la droite vers la gauche.

À travers ces deux dernières fonctions à segmentation et traduction aléatoires, nous pouvons remarquer que des segments courts peuvent produire des gènes (paires de segments) encodant des portions intéressantes de la traduction complète e. En effet, les longs segments même s'ils couvrent plus de contraintes de traduction, peuvent entraîner une perte de qualité sur la globalité de l'hypothèse de traduction une fois les segments concaténés. En TAS à base de segments, ce problème est connu sous le nom de *"problème des frontières entre segments"*. Afin de produire une bonne traduction complète, il faut trouver les bonnes frontières entre les segments sources pour que leur segments cibles une fois concaténés produisent la bonne traduction. C'est la fonction d'évaluation des chromosomes (*fitness*) basée sur l'approche log-linéaire combinant plusieurs scores, qui va guider le décodeur afin de trouver le bon équilibre entre segments longs et courts.

3.4 Croisement

Le croisement est la fonction qui permet d'accoupler deux chromosomes de la population afin de produire à partir de ceux-ci deux nouveaux chromosomes enfants, ou plusieurs nouveaux chromosomes enfants si nous l'appliquons plusieurs fois sur le même couple de parents. Le mécanisme de croisement à mettre en place doit produire de nouveaux chromosomes qui respectent l'encodage défini auparavant. Dans GAMaT, le croisement ne doit pas produire des chromosomes qui ne font pas partie de l'espace de recherche de la phrase source f et cela en respectant les contraintes de l'encodage d'un chromosome. En effet, un chromosome dans GAMaT, doit toujours avoir le même nombre de segments sources que de segments cibles et des alignements de type 1-1 entre ces segments pour en former des paires. Ces deux contraintes doivent être toujours

respectées, car nous travaillons dans le cadre de la TAS à base de segments et non pas à base de mots (Voir Section 1.4.2 du Chapitre 1), dans laquelle il n'y a que des alignements de type 1-1, où chaque segment source doit avoir un segment cible.

Comme nous l'avons présenté dans la section précédente, une solution, dans GAMaT, n'est pas qu'une simple suite de valeurs numériques, mais un ensemble d'éléments de différents types et de différentes tailles. Cette structure du chromosome dans GAMaT, nous a contraint à proposer un nouveau mécanisme de croisement différant de ce qui existe dans la littérature.

La fonction de croisement que nous avons implémentée prend en entrée deux chromosomes parents cp^i et cp^j parmi les chromosomes de la population sélectionnés pour la reproduction. L'idée du croisement est de sélectionner une portion de chacun des deux chromosomes parents et les interchanger pour créer deux nouveaux chromosomes enfants ce^i et ce^j. Une portion d'un chromosome est constituée d'une suite de paires de segments sources et de leur segments cibles.

Les deux portions sélectionnées, ont la même partie source $f_{début}^{fin}$, où *début* est la position du premier mot de la portion dans la phrase source f et fin est la position du dernier mot de la portion dans la phrase source. La sélection de ces deux portions est faite de manière aléatoire sous deux conditions :

— le premier mot source $f_{début}$ des deux portions doit être le début de segment dans les deux chromosomes parents cp^i et cp^j.
— le dernier mot source f_{fin} des deux portions doit être la fin de segment dans les deux chromosomes parents cp^i et cp^j.

Dans la figure 3.8, nous présentons un exemple de ce croisement entre deux chromosomes qui encodent deux hypothèses de traduction différentes pour la même phrase source. Nous remarquons que les deux chromosomes n'ont pas la même segmentation ni le même nombre de segments.

Dans la figure 3.8, nous pouvons voir que *"la maison blanche des USA"* est la partie source des deux portions sélectionnées dans les deux chromosomes parents. Le premier mot $f_{début}$= *"la"* est le début du premier segment $cp^1.\overline{f}_1$ du chromosome parent cp^1 et le début du premier segment $cp^2.\overline{f}_1$du chromosome cp^2. Le dernier mot f_{fin}= *"USA"* est la fin du deuxième segment $cp^1.\overline{f}_2$ du chromosome parent cp^1 et la fin du troisième segment $cp^2.\overline{f}_3$ du chromosome cp^2. Le croisement consiste à prendre la portion source *"la maison blanche des USA"* avec sa segmentation et les segments cibles alignés du premier chromosome parent cp^1 et l'échanger avec la même portion du deuxième chromosome parent cp^2. À travers cet exemple, nous pouvons voir que les deux chromosomes enfants produits encodent deux nouvelles hypothèses de traduction différentes de celles encodées dans les chromosomes parents. Les chromosomes enfants ont aussi des segmentations différentes et des alignements différents par rapport aux parents.

Pour les mêmes deux chromosomes parents cp^1 et cp^2 de la figure 3.8, nous pouvons appliquer un autre croisement en sélectionnant une autre portion. En effet, si nous prenons la portion ayant comme partie source *"est différente de celle du Maroc"* avec $f_{début}$= *"est"* et f_{fin}=*Maroc"*, alors nous pouvons produire deux autres chromosomes enfants différents de ceux du premier croisement. Le troisième, et dernier, croisement possible pour ces deux chromosomes parents est en sélectionnant la portion ayant comme partie source *"celle du Maroc"*.

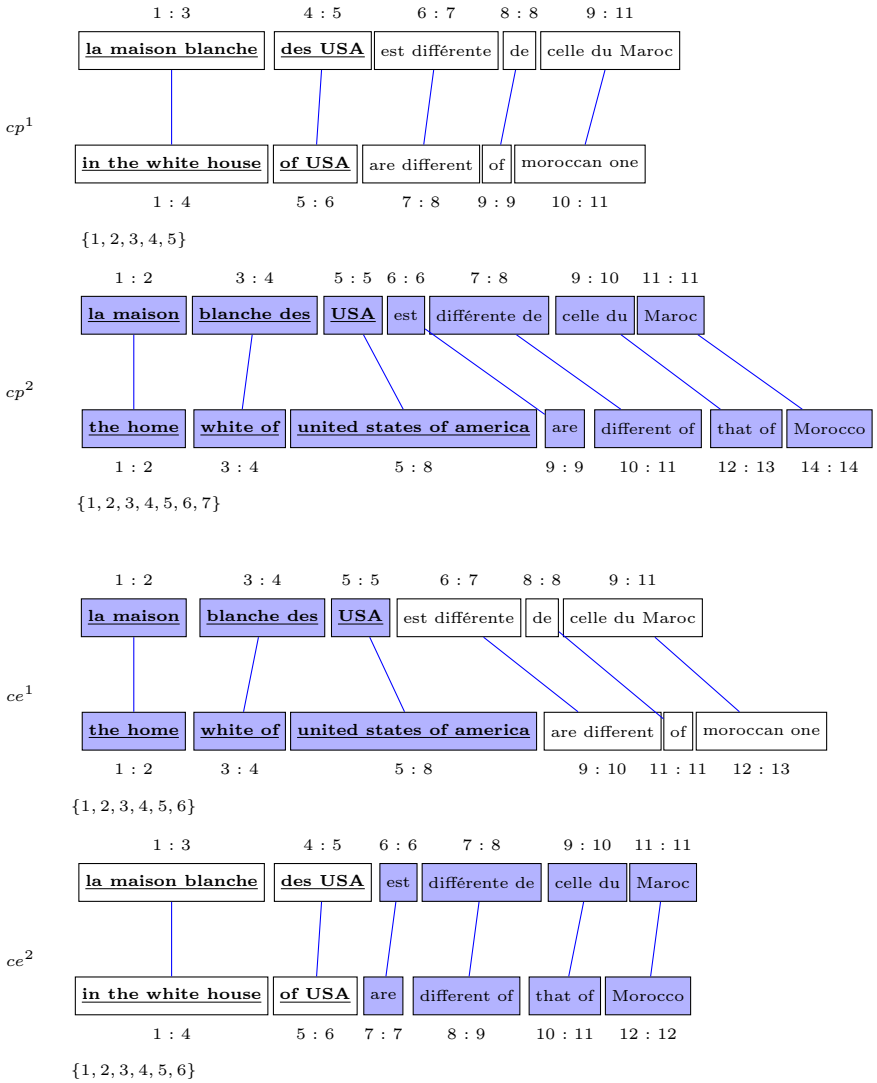


FIGURE 3.8 – Exemple de croisement dans GAMaT.

Nous sommes obligés de mettre en place ces fortes contraintes de croisement afin de s'assurer que les chromosomes enfants résultant du croisement ne soient pas en dehors de l'espace de recherche et afin de garder la cohérence des solutions de la population. Dans la théorie des algorithmes génétiques, la fonction de croisement doit pouvoir croiser n'importe quelle paire de chromosomes de la population, ce qui n'est pas possible avec l'encodage du chromosome dans ce problème.

Dans cette fonction de croisement, si nous ne pouvons pas sélectionner de portions vérifiant ces contraintes, alors le croisement n'est pas appliqué et d'autres chromosomes parents sont sélectionnés. à titre d'exemple, dans les deux chromosomes parents de la figure 3.8, hormis les trois portions que nous avons listées auparavant, aucun autre croisement ne peut être appliqué. Dans la figure 3.9, suivante, nous illustrons un exemple d'un couple de chromosomes parent qui ne peuvent être croisés, car nous ne pouvons trouver de portions vérifiant les deux contraintes.

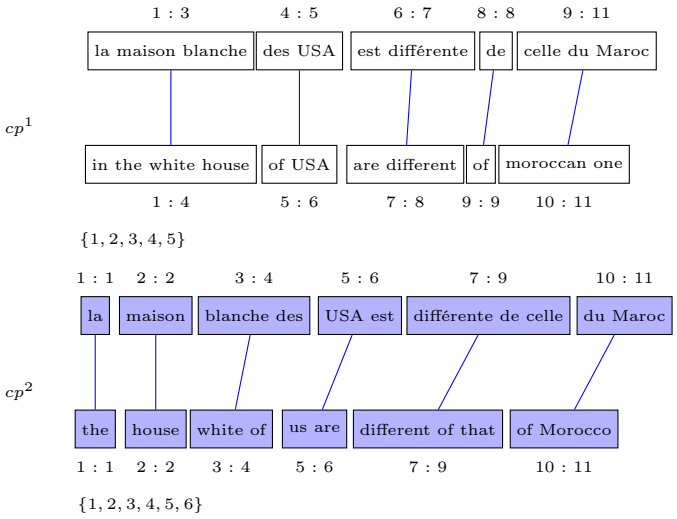


FIGURE 3.9 – Exemple de paire de chromosomes que nous ne pouvons pas croiser dans GAMaT.

Ce mécanisme de croisement permet de produire de nouveaux chromosomes qui se partagent les gènes (paires de segments) des deux parents. Idéalement, un des deux chromosomes enfants encode les meilleurs gènes des deux parents afin de produire une hypothèse de traduction de meilleure qualité que celles des parents.

La fonction de croisement est appliquée un certain nombre de fois sur les chromosomes sélectionnés pour la reproduction parmi ceux de la population. Le taux d'application de la fonction de croisement est un paramètre à fixer de manière empirique (voir le chapitre des expérimentations, Chapitre 5).. Vu que la sélection des deux chromosomes parents se fait de manière aléatoire, nous rajoutons une condition pour qu'un même couple de chromosomes ne soit accouplé qu'une seule fois à chaque itération du processus génétique.

3.5 Mutation

Tout comme pour le croisement, la mutation est appliquée au niveau des paires de segments sources et cibles (gènes). Dans GAMaT, nous proposons 4 fonctions de mutation qui modifient un chromosome de la population de manière différente. Chaque fonction, sélectionne de manière aléatoire un chromosome c^i parmi les chromosomes sélectionnés pour la reproduction et applique une transformation au niveau des gènes pour produire un nouveau chromosome encodant une nouvelle hypothèse de traduction.

Les fonctions de mutation appliquent des transformations génétiques au niveau des gènes et elles sont mises en place pour corriger ou améliorer la qualité des chromosomes de la population et ceux issus du croisement. Idéalement, ces mutations produisent des hypothèses de traduction meilleures que celles qui existaient dans la population.

Dans ce qui suit, nous présentons en détail chacune de ces fonctions de mutation en les illustrant par des exemples.

3.5.1 Remplacement de la traduction

La mutation par remplacement commence par sélectionner aléatoirement un des segments source $\overline{f}_j$ du chromosome c^i. Ensuite, la mutation consiste à remplacer la traduction $\overline{e}_{a_j}$ de ce segment source par une autre à partir de la table de traduction. Cette mutation applique une transformation au niveau de la paire de segments $\overline{f}_j$ et $\overline{e}_{a_j}$.

Dans l'exemple de remplacement de la figure 3.10, nous pouvons voir que la nouvelle traduction *is different* du segment source *"est différente"* est plus correcte d'un point de vue humain et aussi en matière de probabilité de traduction.

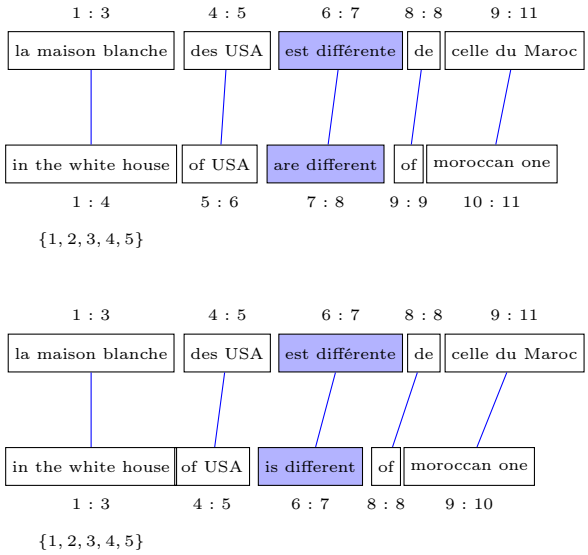


FIGURE 3.10 – Exemple de mutation par remplacement.

La nouvelle traduction $\overline{e}_{a_j}$ est sélectionnée de manière aléatoire parmi les traductions du segment source $\overline{f}_j$ avec une probabilité de traduction plus élevée que la traduction remplacée. Cette stratégie de choix de la nouvelle traduction est basée sur la maximisation de la probabilité de traduction de l'hypothèse encodée dans le chromosome. Si la traduction précédente est la meilleure dans la TT, alors une traduction est sélectionnée de manière aléatoire sans aucune contrainte.

La mutation par remplacement n'a aucun impact sur la segmentation du chromosome et ne modifie pas le nombre de paires de segments. Cependant, cette mutation peut modifier la taille de l'hypothèse de traduction e encodée dans le nouveau chromosome si le nouveau segment cible est plus long ou plus court que le précédent.

3.5.2 Fusion de segments

Cette mutation est mise en place afin de créer de longs segments dans le chromosome. Comme nous l'avons expliqué dans la section d'initialisation, les longs segments englobent plus de relations de traduction que les segments courts.

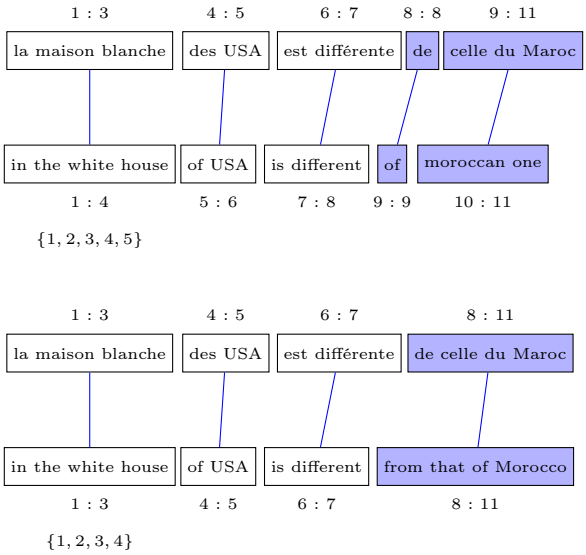


FIGURE 3.11 – Exemple de mutation par fusion.

En pratique, deux segments sources adjacents $\overline{f}_j$ et $\overline{f}_{j+1}$, sont sélectionnés de manière aléatoire dans le chromosome, et fusionnés en un seul segment source $\overline{f}_j$. Le nouveau segment source, résultant de la concaténation, doit exister dans la TT, sinon deux autres segments adjacents sont sélectionnés. Les deux segments cibles associés $\overline{e}_{a_j}$ et $\overline{e}_{a_{j+1}}$ sont fusionnés aussi en un seul segment cible $\overline{e}_{a_j}$. Si les deux segments cibles ne peuvent pas être fusionnés, car leur concaténation n'est pas une traduction (dans la TT) de la concaténation des deux segments sources, alors le nouveau segment source résultant de la fusion est traduit en utilisant la TT.

Dans la figure 3.11, nous pouvons voir que les deux segments sources à fusionner *"de" "celle du Maroc"* dans le chromosome à muter, ont une meilleure traduction une fois fusionnés. La mutation par fusion modifie la segmentation de la phrase source et donc modifie le nombre de paires de segments encodés dans le nouveau chromosome.

3.5.3 Séparation de segment

Cette mutation consiste à sélectionner de manière aléatoire un segment source $cp.\overline{f}_j$ du chromosome parent et le séparer en deux nouveaux segments sources $ce.\overline{f}_j$ et $ce.\overline{f}_{j+1}$ dans le chromosome résultant de la mutation. Les deux nouveaux segments sources doivent exister dans la TT. Le segment source sélectionné doit être composé d'au moins deux mots afin de pouvoir sélectionner un point de séparation de manière aléatoire. Le segment cible associé $\overline{e}_{a_j}$, est aussi séparé en deux segments $\overline{e}_{a_j}$ et $\overline{e}_{a_{j+1}}$. Si les deux nouveaux segments cibles ne peuvent pas être construits à partir de l'ancien segment cible, alors les deux nouveaux segments sources $\overline{f}_j$ et $\overline{f}_{j+1}$ sont traduits en sélectionnant pour chacun sa meilleure traduction dans la TT.

Dans l'exemple de mutation par séparation de la figure 3.12, le segment à muter est *"la maison blanche"* et le point de séparation est le mot *"maison"*. Les deux nouveaux segments sources sont *"la maison"* et *"blanche"*. Dans ce cas, nous ne pouvons pas construire deux nouveaux segments cibles associés aux segments sources à partir du segment cible *"in the white house"*. Les deux nouveaux segments cibles *"the house"* et *"white"* sont les traductions ayant la meilleure

probabilité de traduction des nouveaux segments sources. À la fin de cette mutation, l'ensemble des alignements a est mis à jour afin de garder une cohérence dans le chromosome.

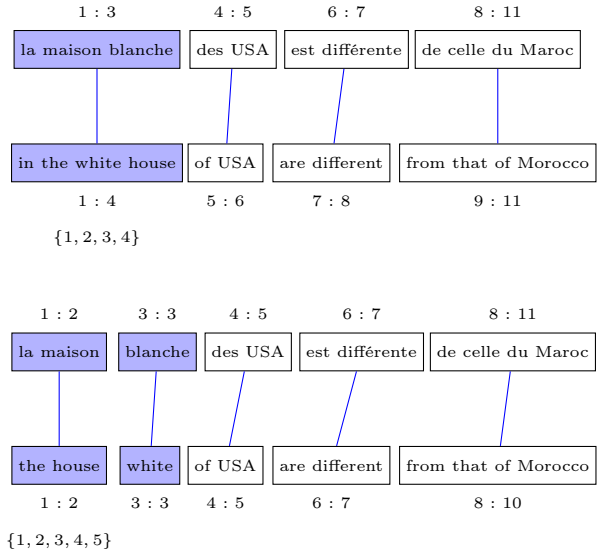


FIGURE 3.12 – Exemple de mutation par séparation.

Nous avons proposé la mutation par fusion, ainsi que la mutation par séparation, dans le but de corriger les erreurs de segmentation qui peuvent exister dans certains chromosomes. En effet le problème de détection de frontières entre les segments est crucial afin de produire une bonne hypothèse de traduction.

3.5.4 Échange de positions de segments

La dernière fonction de mutation est celle qui permet de gérer le réordonnancement dans la cible e. En effet, les mots ou les segments dans la cible n'ont pas forcément le même ordre que leurs mots ou segments alignés dans la source. Le niveau de réordonnancement à appliquer à la cible est en fonction de la paire de langues traitée, où ce réordonnancement varie d'un alignement totalement monotone pour des paires de langues qui se rapprochent dans leurs structures (français-italien), à un alignement fortement réordonné pour des langues très différentes (français-japonais).

Les fonctions d'initialisation, que nous avons présentées dans la Section 3.3, ne produisent que des chromosomes ayant des alignements monotones des paires de segments. Les autres fonctions de croisement et de mutation appliquent des transformations au niveau des paires de segments, en modifiant, pour certaines, la segmentation, mais ne modifient pas l'ordre des segments cibles. Avec cette fonction de mutation, nous corrigeons les problèmes de réordonnancement dans la cible en modifiant les positions des segments cibles afin d'avoir le bon ordre qui permet de produire la bonne hypothèse de traduction une fois les segments cibles concaténés.

Comme illustré dans la figure 3.13, deux segments sources adjacents $cp.\overline{f}_j$ et $cp.\overline{f}_{j+1}$ sont sélectionnés de manière aléatoire dans le chromosome cp à muter. Cette mutation modifie l'alignement a du chromosome, où nous aurons en résultat, dans le chromosome ce qui résulte de la

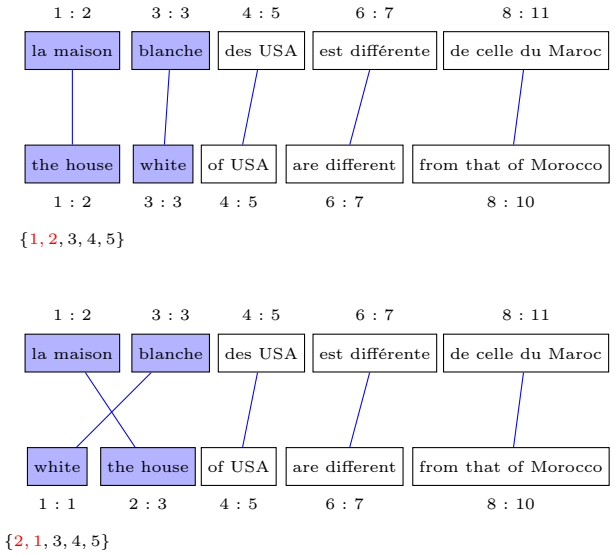


FIGURE 3.13 – Exemple de mutation par échange de positions.

mutation, $ce.a_j = cp.a_{j+1}$ et $ce.a_{j+1} = cp.a_j$. Dans l'exemple, nous pouvons voir que l'échange des positions des deux segments cibles *"the house"* et *"white"* permet de proposer une hypothèse de traduction qui ne semble pas meilleure que la précédente, mais avec une autre mutation de type remplacement de traduction appliquée au segment source *"la maison"*, une hypothèse nettement meilleure peut être produite.

Comme nous l'avons illustré dans les exemples de mutations et de croisement, ce n'est pas une seule transformation qui va produire la bonne hypothèse de traduction. La combinaison des différentes fonctions de mutation avec celle du croisement au cours des générations successives permet d'améliorer la qualité de la population des solutions en produisant des chromosomes encodant des gènes de meilleure qualité que ceux des générations précédentes.

Afin de s'assurer que le processus génétique de GAMaT arrive à faire la différence entre les bonnes hypothèses de traduction et les moins bonnes ou mauvaises, une fonction d'évaluation des chromosomes doit être définie. Cette fonction *fitness* doit évaluer la qualité des hypothèses de traduction encodées dans les chromosomes et guider les processus de sélection et de remplacement. Dans le point qui va suivre, nous détaillons la fonction *fitness* que nous avons implémentée dans GAMaT.

3.6 Fonction d'évaluation *"fitness"*

Dans notre décodeur génétique pour la TAS à base de segments, nous utilisons l'approche log-linéaire présentée dans la Section 1.4.4 du Chapitre 1. L'approche log-linéaire permet de combiner plusieurs scores estimés à partir de différents modèles afin d'évaluer plusieurs aspects des hypothèses de traduction encodées dans les chromosomes de la population.

Dans GAMaT, afin d'évaluer la qualité d'un chromosome c^j de la population, nous utilisons l'approche log-linéaire comme suit :

$$score(c^j) = \sum_{i=1}^{|\lambda|} \lambda_i \times \log h_i(c^j) \tag{3.2}$$

En utilisant la formule 3.2, nous combinons 8 scores, soit estimés à partir de modèles statistiques, soit calculés directement en fonction de la structure du chromosome. Les 8 scores que nous combinons sont listés comme suit :

— h_1 : score du modèle de langue.
— h_2 : score du modèle de traduction direct.
— h_3 : score du modèle de traduction inverse.
— h_4 : score du modèle lexical direct.
— h_5 : score du modèle lexical traduction inverse.
— h_6 : score de réordonnancement.
— h_7 : pénalité de la taille de la traduction.
— h_8 : pénalité du nombre de segments.

Comme nous pouvons le voir dans la formule 3.2, à chaque score h_i est associé un poids λ_i afin de normaliser la combinaison et donner un poids d'importance à chaque score par rapport aux autres dans l'évaluation de la qualité du chromosome. Les valeurs des poids λ doivent être optimisées en fonction de la paire de langues à traiter et cela sur un ensemble de développement (d'optimisation). Dans la littérature, il existe différents algorithmes d'optimisation des poids que nous avons présentés dans la Section 1.4.5 du Chapitre 1. Dans notre travail de thèse, nous avons proposé deux nouvelles approches : un algorithme génétique pour l'optimisation des valeurs de ces poids ; et une approche neuronale pour la combinaison des 8 scores. Dans le chapitre suivant nous présentons en détail ces deux propositions.

Dans les points qui vont suivre, nous détaillons chacun des 8 scores que nous combinons dans l'approche log-linéaire pour évaluer les chromosomes.

3.6.1 Score du modèle de langue

Le score du modèle de langue et celui du modèle de traduction sont les deux scores de base dans l'évaluation des hypothèses de traduction durant le décodage des systèmes de TAS. Le score du modèle de langue, comme présenté en détail dans la Section 1.4.3 du Chapitre 1, est calculé comme suit :

$$h_1(c^j) = P(c^j.e) = \prod_{i=1}^{|c^j.e|} P(c^j.e_i | c^j.e_1, ..., c^j.e_{i-1}) \tag{3.3}$$

Où e_i représente le $i^{éme}$ mot de la traduction e. La probabilité $P(e_i|e_1, \ldots, e_{i-1})$ représente la probabilité d'avoir le mot e_i dans une traduction sachant la suite de mots qui le précède est $e_1, \ldots, e_{i-1}$. De cette manière, plus l'hypothèse de traduction est composée de suites de mots fréquentes dans la langue cible, plus elle a une meilleure probabilité d'être syntaxiquement correcte. Pour les expérimentations, nous utilisons un modèle de langue 5-grammes. Les modèles 5-grammes ont les meilleures performances en TAS.

3.6.2 Score du modèle de traduction direct

Le score de traduction direct d'un chromosome c^j est calculé sur la base des probabilités de traduction des paires de segments encodées dans le chromosome :

$$h_2(c^j) = \prod_{i=1}^{|c^j.a|} P(c^j.\overline{f}_i | c^j.\overline{e}_{a_i}) \tag{3.4}$$

Comme illustré dans la Table 3.1, le probabilité de traduction de chaque paire de segment $c^j.\overline{f}_i$ et $c^j.\overline{e}_{a_i}$ est sauvegardée dans la TT. Comme présenté dans la Section 1.4.2 du Chapitre 1, le score du modèle de traduction direct estime la probabilité d'avoir la phrase source $c^j.f$ sachant que l'hypothèse de traduction est $c^j.e$ et sachant la segmentation et les paires de segments encodées dans le chromosome. Plus cette probabilité est élevée meilleure est l'hypothèse de traduction.

3.6.3 Score du modèle de traduction inverse

Ce score est estimé de la même manière que le score du modèle de traduction direct, sauf que dans ce cas, elle estime la probabilité d'avoir l'hypothèse de traduction $c^j.e$ sachant que la phrase source est $c^j.f$ et sachant aussi, la segmentation et les paires de segments encodées dans le chromosome :

$$h_3(c^j) = \prod_{i=1}^{|c^j.a|} P(c^j.\overline{e}_{a_i} | c^j.\overline{f}_i) \tag{3.5}$$

Les probabilités de traduction inverses sont estimées durant l'apprentissage du modèle de traduction, où le décompte fréquentiel est fait dans les deux sens. GIZA++, outil de référence pour la création des modèles de traduction, sauvegarde la probabilité de traduction inverse de chaque paire de segment dans la TT.

3.6.4 Score du modèle lexical de traduction direct

Une autre façon d'estimer la qualité d'une paire de segments source et cible sauvegardée dans la table de traduction, est d'estimer la qualité des liens qui existent entre les mots du segment source et ceux du segment cible [Koehn et al., 2003]. En effet, comme présenté dans la Section 1.4.2 du Chapitre 1, la construction d'un modèle de traduction à base de segment nécessite la construction au préalable, d'un modèle de traduction à base de mots. Dans ce dernier modèle, des paires de mots sources et cibles avec leur probabilités de traduction directes et inverses sont sauvegardées.

Au moment de la création de la TT à base de segments, pour chaque paire de segments source Sf et cible Se, avec un alignement aw entre les mots de ces segments, la probabilité du modèle lexical direct $P_w(Sf|Se, aw)$ est calculée comme suit [Koehn et al., 2003], [Chiang et al., 2011] :

$$P_w(Sf|Se, a) = \prod_{i=1}^{|Sf|} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{(i,j)\in a} w(Sf_i|Se_j) \tag{3.6}$$

Dans la formule 3.6, $|\{j|(i,j) \in a\}|$ représente le nombre de mots du segment cible alignés au mot source Sf_i et $w(Sf_i|Se_j)$ représente le poids lexical entre le mot source Sf_i et le mot cible Se_j estimé à partir du corpus parallèle et qui est calculé comme suit :

$$w(Sf_i|Se_j) = \frac{count(Sf_i, Se_j)}{count(Se_j)} \tag{3.7}$$

Où $count(Sf_i, Se_j)$ est le nombre de fois où apparaît le mot source Sf_i dans une phrase source et que le mot cible Se_j apparaît dans sa phrase cible alignée. $count(Se_j)$ est le nombre de phrases sources dans lesquelles apparaît le mot cible Se_j.

De la même manière que les deux scores de traduction direct et inverse, le score du modèle lexical direct est calculé comme suit :

$$h_4(c^j) = \prod_{i=1}^{|c^j.a|} P_w(c^j.\overline{f}_i|c^j.\overline{e}_{a_i}) \tag{3.8}$$

3.6.5 Score du modèle lexical de traduction inverse

Le score du modèle lexical de traduction inverse est le quatrième score de traduction. Ce score, estime la qualité des liens qui existent entre les mots des segments cibles et ceux des segments sources, inversement au score précédent. Le score du modèle lexical de traduction inverse est donc calculé comme suit :

$$h_5(c^j) = \prod_{i=1}^{|c^j.a|} P_w(c^j.\overline{e}_{a_i}|c^j.\overline{f}_i) \tag{3.9}$$

3.6.6 Score de réordonnancement

Ce score estime le coût de réordonnancement des segments sources par rapport aux segments cibles pris dans l'ordre [Koehn et al., 2007]. Ce coût varie de 0, où l'alignement est monotone entre les segments sources et cibles, à une valeur maximale, où l'ordre des segments cibles est totalement inversé par rapport l'ordre des segments sources. Cette valeur maximale dépend du nombre de segments dans le chromosome et des tailles de ces segments.

L'importance de la valeur de ce score varie en fonction de la paire de langues traitée. Pour des paires de langues similaires nécessitant peu de réordonnancement, un faible score de réordonnancement est préférable, et inversement pour les paires de langues avec une forte différence. D'un autre côté, le poids λ_6 fixe l'importance de ce score par rapport aux autres scores dans la combinaison.

Pratiquement ce score $h_6(c^j)$ est calculé comme suit :

$$h_6(c^j) = \sum_{i=1}^{|c^j.a|} |(x + 1 - y)| \tag{3.10}$$

Dans cette formule les segments cibles sont pris dans l'ordre, où pour chaque segment cible $\overline{e}_i$, x représente la position, dans la phrase source f, du dernier mot du segment source aligné au

segment cible $\overline{e}_{i-1}$, celui qui précède le segment $\overline{e}_i$. Quant à y, il représente la position, dans la phrase source f, du premier mot du segment source aligné au segment cible courant $\overline{e}_i$.

Le calcul de ce score pour le chromosome c de l'exemple précédent (voir Figure 3.13) se fait comme suit :

— $h_6(c)$= |0+1-3| + |3+1-1| + |2+1-4| + |5+1-6| + |7+1-8| + |8+1-9| = 2+3+1+0+0+0 = 6

Dans l'exemple, il n'y a qu'un seul échange de position entre les segments sources et cibles. L'échange est au niveau des deux premiers segments. Ce réordonnancement dans le chromosome, influence le coût de réordonnancement des trois premiers segments cibles. Cependant, pour les 4 derniers segments de la cible qui ont un alignement monotone, le coût à leur niveau vaut 0. Nous pouvons voir que la valeur de ce score dépend d'un côté du réordonnancement au niveau des segments, et d'un autre côté des distances entre les mots de début et de fin des segments réordonnés.

3.6.7 Pénalité de la taille de l'hypothèse de traduction

La pénalité de la taille de l'hypothèse de traduction est calculée comme étant la valeur exponentielle de la taille de l'hypothèse de traduction e encodée dans le chromosome c^j [Koehn, 2004], [Zens et al., 2002], [González, 2012] :

$$h_7(c^j) = \exp(|c^j.e|) \tag{3.11}$$

Pour le chromosome de l'exemple précédent, la pénalité de la taille de l'hypothèse de traduction est égale à $\exp(10) = 27.18281$.

Ce score est ajouté à la combinaison log-linéaire afin d'estimer la qualité de l'hypothèse de traduction en termes de nombre de mots. En effet, le score du modèle de langue a tendance à favoriser les courtes hypothèses de traduction car elles ont des probabilités plus élevées que les longues hypothèses. Afin de minimiser l'impact de la taille de l'hypothèse et éviter la convergence vers les courtes traductions, cette pénalité est ajoutée pour contrebalancer la probabilité du modèle de langue.

3.6.8 Pénalité du nombre de segments

Comme pour le score précédent, la pénalité du nombre de segments est ajoutée à la combinaison pour contrebalancer les scores des 4 modèles de traduction qui ont tendance à être élevés pour un faible nombre de segments, donc ils favorisent les chromosomes avec de faibles nombres de segments. Dans la littérature, La pénalité de chaque paire de segments est de $e = 2.718$ et la pénalité sur l'ensemble du chromosome est estimée comme suit [Koehn, 2004], [Zens et al., 2002], [González, 2012] :

$$h_8(c^j) = \prod_{i=1}^{|a|} e = e^{|a|} \tag{3.12}$$

La pénalité du nombre de segments d'un chromosome est donc le produit k fois e, k étant le nombre de paires de segments dans le chromosome.

3.7 Sélection et remplacement

Comme nous l'avons présenté dans la Section 3 du Chapitre 2, les deux mécanismes de sélection et de remplacement assurent, à travers la fonction *fitness*, la bonne convergence de la population vers une population de chromosomes de bonne qualité. La sélection consiste à sélectionner certains chromosomes d'une population pour leur permettre de se reproduire *via* le croisement et les mutations. Quant au remplacement, il est appliqué à la fin de chaque itération du processus itératif de recherche, et il consiste à garder n chromosomes de la population pour l'itération suivante et éliminer le reste des chromosomes.

Dans GAMaT [Douib et al., 2016], nous implémentons deux stratégies pour les deux mécanismes de sélection et de remplacement que nous détaillons comme suit :

— La première stratégie est une sélection ordinale basée sur la *fitness* des chromosomes de la population. Au début de chaque itération, les k chromosomes les mieux évalués, parmi les n chromosomes de la population, sont sélectionnés pour être croisés et mutés afin de produire m nouveaux chromosomes. De la même manière, à la fin de chaque itération, les n chromosomes les mieux évalués parmi les $n+m$ chromosomes de la population et ceux résultant de la reproduction, sont gardés afin de produire une nouvelle population pour l'itération suivante.

— La deuxième stratégie est basée sur une sélection par roulette. La même stratégie est appliquée pour la sélection des chromosomes pour la reproduction ainsi que pour le remplacement à la fin de chaque itération. Afin d'adapter cette stratégie à l'algorithme génétique de GAMaT nous procédons comme suit :

1. Évaluer l'ensemble des chromosomes et détecter la meilleure valeur *fitness* max_{score} ainsi que la plus faible min_{score}.
2. Créer 4 intervalles, de même taille, de valeurs de *fitness*. Le premier intervalle étant $[min_{score}; min_{score} + pas[$, et le dernier étant $[max_{score} - pas; max_{score}[$. La taille de ces 4 intervalles est calculée comme suit : $pas = (max_{score} - min_{score})/4$.
3. Associer à chaque intervalle une valeur de sélection entre 0 et 100. L'intervalle des valeurs *fitness* élevées a une valeur de sélection de 60, la plus forte des 4 intervalles. Les deuxième et troisième intervalles ont, respectivement, des valeurs de sélection de 20 et 15. Le dernier intervalle, des valeurs *fitness* les plus faibles, a une valeur de sélection égale à 5.
4. Classer l'ensemble des chromosomes de la population dans les 5 intervalles en fonction de leur *fitness*.
5. Lancer la roulette le nombre de fois qu'il faut pour la sélection ou le remplacement. Un lancer correspond à la génération d'une valeur entre 0 et 100 de manière aléatoire. Si la valeur est inférieure à 60, alors sélectionner aléatoirement un chromosome de l'intervalle des chromosomes les mieux évalués. Si la valeur est entre 60 et 80 alors sélectionner un chromosome parmi ceux du deuxième intervalle. Si cette valeur est entre 80 et 95, alors sélectionner un chromosome du troisième intervalle. Sinon, sélectionner un chromosome parmi ceux qui ont une faible *fitness* (quatrième intervalle).

En pratique, nous avons expérimenté 3 scénarios combinant ces deux stratégies sur les deux

mécanismes de sélection et de remplacement. Dans le premier scénario, nous utilisons une sélection ordinale et un remplacement ordinal. Le deuxième, une sélection à base de roulette et un remplacement à base de sélection ordinale. Dans le troisième scénario, nous utilisons la stratégie de roulette pour la sélection ainsi que pour le remplacement. Dans le chapitre des expérimentations de ce manuscrit, nous présentons des résultats comparatifs entre ces 3 scénarios.

Dans les deux mécanismes de sélection et de remplacement, le meilleur chromosome ayant la *fitness* la plus élevée est automatiquement sélectionné afin de ne pas perdre la meilleure solution d'une itération à une autre à cause des décisions semi-aléatoires de ces deux mécanismes. D'un autre côté, pour éviter une convergence rapide vers une région restreinte de l'espace de recherche, nous éliminons les chromosomes en double en vérifiant avant la sélection de chaque chromosome qu'il n'y a pas un chromosome égal déjà sélectionné.

3.8 Déroulement du processus génétique de GAMaT

De manière globale, le processus génétique de GAMaT a un déroulement similaire à celui des algorithmes génétiques présenté dans la Section 2.3 du Chapitre 2. Dans GAMaT, nous avons adapté les différents composants et mécanismes des AG au problème de la TAS à base de segments comme nous l'avons présenté dans ce chapitre. Nous illustrons, de manière détaillée, ce processus génétique de GAMaT dans la figure 3.14.

À travers la figure 3.14, nous pouvons voir que certains mécanismes du processus génétique sont appliqués plusieurs fois et de manière itérative. En effet, le croisement et les fonctions de mutation sont appliqués plusieurs fois afin de produire plusieurs nouveaux chromosomes. Le nombre de fois que nous appliquons ces fonctions de reproduction dépend des taux de croisement et de mutation. Ces deux taux sont fixés de manière empirique à condition d'avoir un taux de croisement toujours supérieur au taux de mutation [Holland, 1973], [Zogheib, 2011] pour simuler le processus naturel d'évolution d'une espèce dans lequel les accouplements sont plus fréquents que les mutations. Contrairement à la littérature des AG [Holland, 1973], nous ne limitons pas l'application des mutations qu'aux chromosomes résultant du croisement mais nous les appliquons à n'importe quel chromosome parmi les k chromosomes sélectionnés pour la reproduction.

Le processus itératif de GAMaT qui englobe l'évaluation, la sélection, le croisement, les mutations et le remplacement, continue de manière itérative jusqu'à une condition d'arrêt. Nous utilisons deux critères pour arrêter ce processus itératif :

— La convergence de la population de chromosomes vers une région optimale. La convergence consiste à atteindre une population que ne nous pouvons plus améliorer. Pour vérifier cela, nous calculons la somme des valeurs *fitness* des chromosomes de la population et si cette somme ne s'améliore pas durant 10 itérations (générations) consécutives alors le processus est stoppé.
— Le processus génétique itératif est stoppé de manière brusque si le nombre d'itérations dépasse un certain seuil. Nous fixons ce seuil à un niveau très élevé (1000 itérations) pour laisser le processus génétique le temps de converger. Dans le chapitre des expérimentations nous présentons des performances en termes de nombre d'itérations.

Dans notre décodeur génétique GAMaT plusieurs paramètres sont à fixer afin d'ajuster le pro-

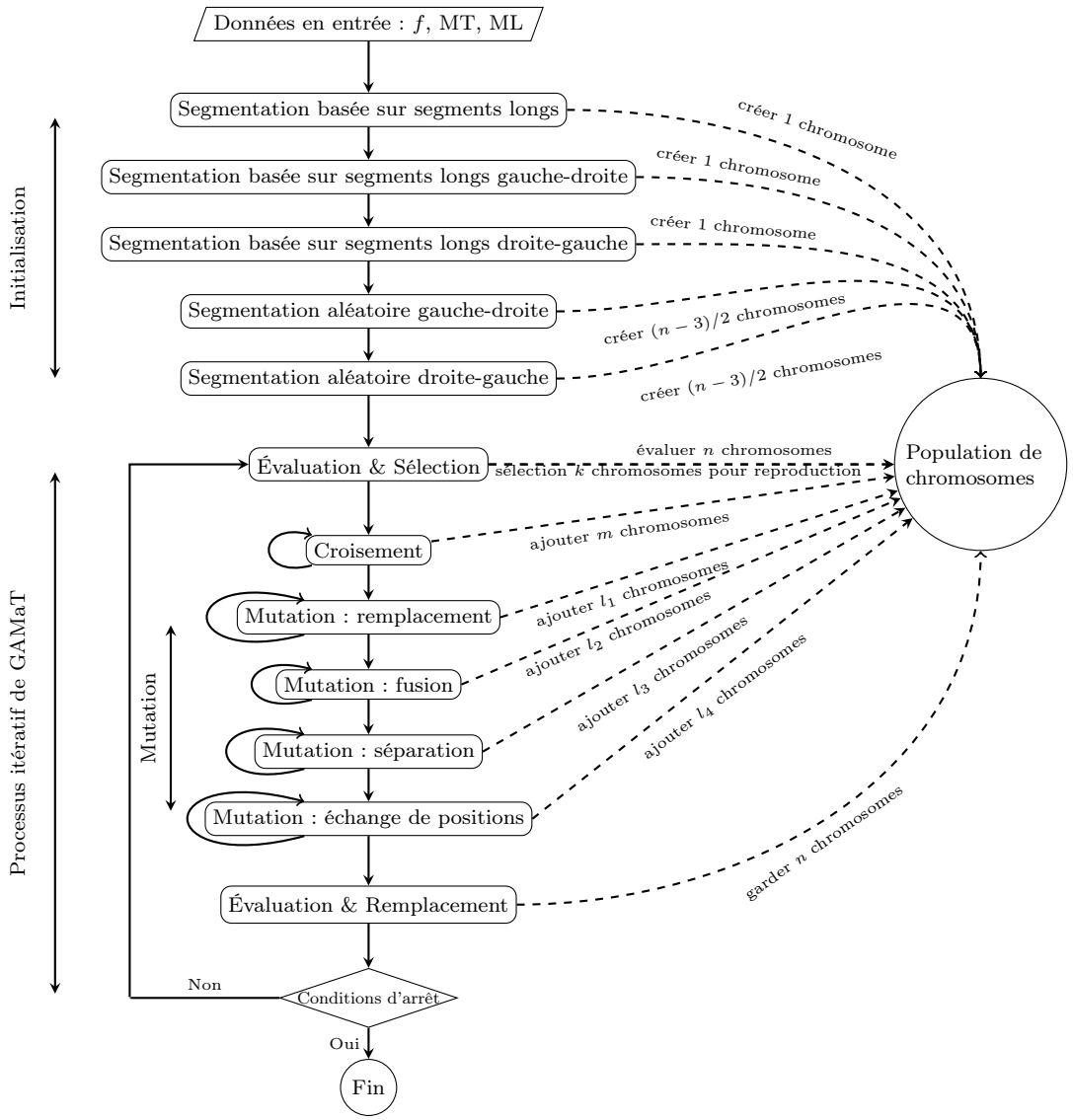


FIGURE 3.14 – Processus génétique détaillé de GAMaT.

cessus génétique. à travers la figure 3.14, nous pouvons voir qu'il y a différents paramètres que nous listons comme suit :

— taille de la population n
— nombre de chromosomes pour la reproduction k
— nombre de chromosomes à produire par croisement m
— nombre de chromosomes à produire par mutations $l1 + l2 + l3 + l4$
— nombre d'itérations global
— nombre d'itérations de stagnation

Dans le chapitre des expérimentations, nous présentons les performances de notre décodeur génétique GAMaT tout en justifiant le choix des valeurs pour ces différents paramètres.

3.9 Conclusion et discussion

Dans ce chapitre, nous avons présenté le cœur de notre travail de thèse, à savoir GAMaT, le décodeur génétique pour la traduction automatique statistique à base de segments. Ce décodeur a l'avantage de manipuler un ensemble d'hypothèses de traduction (population de solutions) tout au long du processus de décodage. Nous avons essayé de diversifier l'ensemble de chromosomes grâce aux fonctions d'initialisation qui permettent de produire des chromosomes encodant différents gènes. Cette diversification permet d'améliorer le processus d'exploration de l'espace de recherche et de visiter un grand nombre de régions. Quant aux mécanismes de reproduction que nous avons mis en place, ils permettent, d'un côté, d'exploiter le potentiel d'une population de chromosomes en y appliquant des transformations génétiques successives, au niveau des paires de segments, à travers les différentes générations. Ces transformations successives arrivent à créer de nouveaux chromosomes encodant des gènes de bonne qualité une fois concaténés, qui existaient dans différents chromosomes dans les générations précédentes. D'un autre côté, ces mécanismes de reproduction permettent d'explorer l'espace de recherche en produisant des nouveaux gènes qui n'existaient pas auparavant en particulier grâce aux fonctions de mutation.

La structure des chromosomes de GAMaT permet d'évaluer des hypothèses de traduction complètes et non pas partielles qui peuvent évoluer, par la suite, positivement ou négativement. Avec cette représentation des solutions, nous espérons mettre en place une fonction d'évaluation (fonction *fitness*) plus efficace, qui facilite les prises de décisions au cours du décodage. En effet, le score du modèle de langue ou ceux des modèles de traduction dépendent fortement de la taille de l'hypothèse et du nombre de segments. Donc les valeurs de ces scores sur des hypothèses partielles ne donnent pas une information sur la qualité globale de la solution mais sur une portion qui peut se détériorer et ou s'améliorer après extension. Dans GAMaT, les prises de décisions au niveau de la sélection et du remplacement sont basées sur des scores de *fitness* d'hypothèses complètes et par conséquent les bonnes solutions ne sont pas perdues au cours du processus de recherche à condition d'avoir une fonction d'évaluation bien optimisée.

Dans un algorithme génétique, la définition d'une fonction d'évaluation de chromosomes efficace est un point très important pour assurer la bonne convergence de la population et atteindre la solution optimale. Dans le chapitre suivant, nous présentons nos deux propositions pour la fonction d'évaluation des chromosomes dans GAMaT. La première proposition est un algorithme génétique pour l'optimisation des poids λ de la fonction log-linéaire afin d'ajuster la

fonction d'évaluation et améliorer le processus de prise de décision. La deuxième proposition est l'utilisation d'un réseau de neurones dans le but d'apprendre une fonction combinant les 8 scores des différents modèles et prédire une valeur estimant la qualité de l'hypothèse encodée dans le chromosome.

Dans le chapitre des tests et expérimentations, nous présentons différents résultats et performances de traduction de notre système GAMaT. Différentes expérimentations sont réalisées afin de comparer les performances de GAMaT en utilisant les différentes approches d'optimisation de la combinaison des 8 scores d'évaluation.

4

Fonction *fitness* & Optimisation des poids de l'approche log-linéaire

4.1 Introduction

Dans la théorie des algorithmes génétiques [Holland, 1973], [Hüe, 1997], la fonction *fitness*, qui évalue la qualité des chromosomes (solutions), est la fonction qui guide tout le processus génétique et qui assure la bonne convergence du processus de recherche vers une population de solutions, qui doit contenir la solution optimale finale. Comme nous l'avons présenté dans la Section 2.3 du Chapitre 2, la fonction *fitness* doit être définie en fonction du problème d'optimisation traité afin d'évaluer la qualité des chromosomes d'une population. La valeur *fitness* d'un chromosome représente à quel point la solution encodée dans ce chromosome arrive à résoudre le problème d'optimisation.

Dans GAMaT, le décodeur génétique pour la traduction automatique statistique à base de segments, un chromosome encode plusieurs éléments comme l'hypothèse de traduction, la segmentation de la source et de la cible et les alignements entre segments sources et cibles (voir Section 3.2 du Chapitre 3). Tous ces éléments doivent être évalués afin d'affecter un seul score estimant la qualité du chromosome. Dans la littérature de la TAS, plusieurs scores estimant différents aspects d'une solution, sont combinés afin d'évaluer la qualité de la solution. Dans GAMaT, pour évaluer les chromosomes, nous combinons 8 scores considérés comme étant les scores de base dans l'état de l'art [Koehn et al., 2003], [Chiang et al., 2011] :

- s_1 : score du modèle de langue.
- s_2 : score du modèle de traduction direct.
- s_3 : score du modèle de traduction inverse.
- s_4 : score du modèle lexical direct.
- s_5 : score du modèle lexical traduction inverse.
- s_6 : score de réordonnancement.
- s_7 : pénalité de la taille de la traduction.
- s_8 : pénalité du nombre de segments.

Dans le chapitre précédent (voir Section 3.6 du Chapitre 3), nous avons présenté chacun de ces 8 scores en détail.

Parmi ces 8 scores, certains sont des vraisemblances, comme le score du modèle de langue ou les scores des modèles de traduction. D'autres scores sont calculés directement en fonction de la structure du chromosome. Le fait que ces scores évaluent différents aspects de la solution et de différentes manières, fait qu'ils n'ont pas le même ordre de grandeur. En effet, une différence de 0.5, en score du modèle de langue entre deux solutions, représente un écart considérable en matière de qualité de la traduction dans la langue cible. Cependant, si deux solutions ont des traductions composées respectivement, de 9 et 10 mots, la différence en matière de pénalité de la taille de la traduction, est de 2.71, qui est nettement supérieur au 0.5 du modèle de langue. Ce score de 2.71 ne représente finalement qu'une faible différence en matière de taille de la traduction (Voir Section 3.6 du chapitre précédent). D'un autre côté, nous avons vu dans le chapitre précédent, qu'il y avait entre certains scores un lien direct, comme pour la probabilité du ML et la pénalité de la taille de l'hypothèse ou encore, entre les probabilités des modèles de traduction et la pénalité du nombre de segments. La mise en place d'une stratégie de combinaison permettant de prendre en considération ces différences et ces liens qui peuvent exister entre les différents scores est nécessaire afin d'assurer la bonne convergence du processus de recherche génétique de GAMaT.

Dans la littérature de la TAS [Och, 2002], [Koehn et al., 2003], au niveau du décodeur, l'approche log-linéaire est utilisée pour définir une fonction d'évaluation multicritères combinant les 8 scores (voir Section 1.4.4 du Chapitre 1). Dans cette approche, des poids de pondération sont associés aux différents scores afin de fixer le niveau d'importance de chaque score par rapport aux autres dans la combinaison. L'optimisation des valeurs de ces poids, a pour objectif d'assurer un équilibre entre les différents scores et leur ordre de grandeur.

Les poids associés aux scores doivent être optimisés en fonction de la paire de langues traitée et aussi en fonction des données que nous utilisons. Dans la littérature de la TAS, différents algorithmes d'optimisation des poids de la fonction log-linéaire sont proposés (voir Section 1.4.5 du Chapitre 1) [Och, 2003], [Hasler et al., 2011], [Hopkins and May, 2011], [Kocur and Bojar, 2016]. Cette optimisation consiste à trouver les bonnes valeurs des poids qui permettent au décodeur de produire des traductions qui maximisent un score d'évaluation par rapport aux traductions de référence sur un ensemble de développement.

Dans ce travail de thèse, nous proposons deux approches pour l'optimisation de la combinaison des différents scores d'évaluation pour notre décodeur GAMaT. La première approche consiste en la proposition d'un nouvel algorithme d'optimisation des poids de la fonction log-linéaire basé sur les algorithmes génétiques [Douib et al., 2017a]. Cette approche nous permet de proposer un système de TAS avec un décodeur entièrement construit à base d'algorithmes génétiques. La deuxième approche consiste à remplacer la combinaison log-linéaire par une approche neuronale [Douib et al., 2017b]. L'idée est d'utiliser un réseau de neurones pour la régression afin d'apprendre une fonction qui combine les différents scores et prédit une valeur estimant la qualité de la traduction encodée dans le chromosome. Au moment de l'apprentissage du réseau de neurones, les valeurs de sortie à prédire sont les valeurs d'une métrique d'évaluation de la traduction (BLEU ou TER).

Dans la Section 2 de ce chapitre, nous présentons GAWO l'algorithme génétique pour l'optimisation des poids de la fonction log-linéaire en détaillant toutes les décisions de conception. Dans la Section 3, nous présentons l'approche neuronale pour la définition de la fonction *fitness* pour le décodeur GAMaT. Une fois ces deux approches présentées, nous pourrons parler d'un

système de TAS génétique complet que nous évaluons dans le chapitre suivant.

4.2 GAWO : algorithme génétique pour l'optimisation des paramètres du système de traduction

GAWO *"Genetic Algorithm for Weights Optimization"* est le nom de notre algorithme génétique pour l'optimisation des poids de la fonction log-linéaire utilisée pour l'évaluation des chromosomes au cours du décodage de GAMaT.

Nous considérons un ensemble de développement composé de n phrases sources $f = \{f_1, ..., f_n\}$ et de leur traduction de référence $r = \{r_1, ..., r_n\}$. En utilisant le décodeur de GAMaT avec la fonction log-linéaire, pondérée avec les poids λ, comme *fitness*, nous traduisons les n phrases sources pour produire n traductions en sortie $\hat{e}^{\lambda} = \{\hat{e}_1^{\lambda}, ..., \hat{e}_n^{\lambda}\}$. Le but de l'optimisation est de trouver les valeurs des poids $\hat{\lambda} = \{\hat{\lambda}_1, ..., \hat{\lambda}_8\}$ qui minimisent la valeur de la fonction d'erreur Err. Cette fonction Err est définie pour estimer la différence entre les traductions en sortie du décodeur et les traductions de référence. Ce problème d'optimisation des poids est défini comme suit :

$$\hat{\lambda} = \underset{\lambda}{argmin}\Big[\sum_{i=1}^{n} Err(\hat{e}_i^{\lambda}, r_i)\Big] \tag{4.1}$$

Comme présenté dans la Section 1.4.5 du Chapitre 1, afin de minimiser le nombre de fois que nous exécutons le décodeur et de minimiser la durée d'optimisation, dans GAWO, l'optimisation se déroule à travers deux boucles imbriquées (voir Figure 4.1) que nous présentons dans les points suivants.

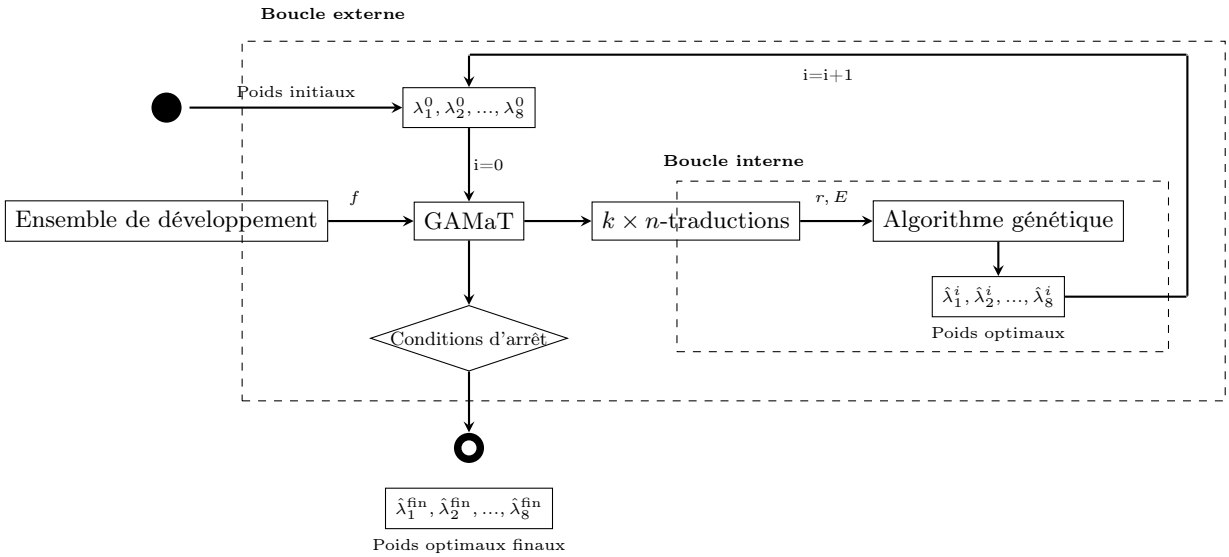


FIGURE 4.1 – Schématisation du processus d'optimisation dans GAWO.

4.2.1 GAWO : processus externe

Dans la boucle externe, nous utilisons GAMaT comme décodeur pour traduire les phrases sources de l'ensemble de développement. Pour chaque phrase source f_i, nous produisons un ensemble de k traductions $E_i = \{e_i^1, ..., e_i^k\}$. Par conséquent, le résultat de cette boucle est un ensemble E de plusieurs ensembles de k traductions $E = \{E_1, ..., E_n\}$. Ces ensembles sont injectés dans la boucle interne pour le processus d'optimisation génétique.

Pour la première itération de la boucle externe, nous utilisons un ensemble initial de poids générés de manière aléatoire $\{\hat{\lambda}_1^0, \hat{\lambda}_2^0, ..., \hat{\lambda}_8^0\}$. Pour les itérations suivantes, l'ensemble de poids obtenu par la boucle interne $\{\hat{\lambda}_1^i, \hat{\lambda}_2^i, ..., \hat{\lambda}_8^i\}$ est utilisé pour exécuter GAMaT afin de retraduire l'ensemble de développement et produire de nouveaux ensembles de k traductions. Le processus des deux boucles, continue de manière itérative et s'arrête lorsque les poids ne peuvent plus être améliorés ou qu'aucune nouvelle traduction ne peut être produite. Les poids optimaux finaux $\{\hat{\lambda}_1^{\text{fin}}, \hat{\lambda}_2^{\text{fin}}, ..., \hat{\lambda}_8^{\text{fin}}\}$ sont utilisés par la suite pour lancer GAMaT afin de traduire des phrases de test et évaluer les performances de traduction de GAMaT.

4.2.2 GAWO : processus interne

Dans la boucle interne, une implémentation classique de l'algorithme génétique est appliquée afin de trouver le jeu de poids optimal qui permet à la fonction d'évaluation log-linéaire de sélectionner le maximum de meilleures traductions parmi les $k \times n$ traductions des phrases sources selon une fonction Err. La fonction Err à minimiser est définie comme étant une métrique d'évaluation de la qualité des traductions [Neubig and Watanabe, 2016], en particulier BLEU [Papineni et al., 2002] et TER [Snover et al., 2006]. Donc, le jeu de poids optimal $\hat{\lambda}$ est celui qui permet d'obtenir un ensemble de n traductions maximisant ou minimisant, selon la métrique, le score d'évaluation des traductions de l'ensemble de développement. Dans la formule 4.2, nous reformulons ce problème d'optimisation en utilisant la métrique BLEU comme fonction Err :

$$\hat{\lambda} = \underset{\lambda}{argmax}\Big[BLEU(\hat{e}^{\lambda}, r)\Big] \tag{4.2}$$

Où $\hat{e}^{\lambda}$ est l'ensemble des n traductions sélectionnées par la fonction log-linéaire avec le jeu de poids λ, et r est l'ensemble des n traductions de référence.

L'idée principale du processus génétique d'optimisation est de commencer avec une population de chromosomes (solutions), c'est-à-dire une population de vecteurs des valeurs des 8 poids, et d'améliorer de manière itérative la qualité de la population en produisant de nouveaux chromosomes. Afin d'assurer la bonne évolution de la population, la fonction de *fitness* est définie comme étant le score BLEU (voir la formule 4.2) pour évaluer les chromosomes et appliquer les processus de sélection et de remplacement. Le processus d'optimisation génétique se poursuit de manière itérative jusqu'à ce qu'il atteigne les meilleurs poids possibles pour les ensembles de traductions actuels $E = \{E_1, ..., E_n\}$.

Dans les points qui vont suivre, nous présentons notre algorithme génétique implémenté dans la boucle interne tout en détaillant chaque composante et les différents choix de stratégies. Nous

détaillons, particulièrement, le processus d'évaluation des solutions (la fonction *fitness*).

Encodage

Dans les algorithmes génétiques, tous les aspects d'une solution sont encodés dans un chromosome, de manière optimisée, afin de pouvoir définir et appliquer les différentes fonctions de reproduction et d'évaluation, comme nous l'avons présenté dans la Section 2.3 du Chapitre 2.

Dans GAWO, une solution est un ensemble de 8 valeurs numériques des poids associés aux 8 scores combinés pour l'évaluation des solutions de GAMaT. Par conséquent, un chromosome (voir Figure 4.2) encode un vecteur de 8 valeurs numériques $c = \{v_1, ..., v_8\}$, où chaque élément v_i représente la valeur du poids λ_i.

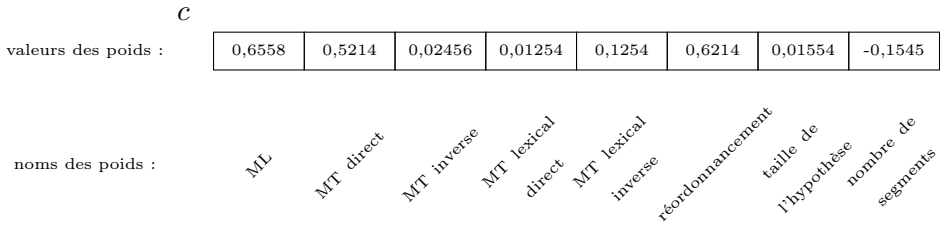


FIGURE 4.2 – Exemple d'un chromosome dans GAWO.

Initialisation

Les chromosomes (solutions) de la première population sont générés de manière aléatoire. Chaque poids prend une valeur aléatoire dans l'intervalle $[-1, 1]$. À cette population initiale générée aléatoirement, nous ajoutons un autre chromosome avec l'ensemble des poids utilisés dans la boucle externe pour lancer GAMaT.

Nous produisons une population de 500 chromosomes, de manière aléatoire. Ce grand nombre de chromosomes aidera à diversifier la population initiale. À l'initialisation, nous évitons d'avoir deux chromosomes identiques pour éviter une convergence rapide vers un minimum local et afin de maximiser les chances d'exploration de l'espace de recherche.

Une autre stratégie d'initialisation que nous testons est de fixer des contraires sur les valeurs des différents poids. Ces contraintes sont en fonction de l'importance, *a priori*, de certains scores par rapport à d'autres. En effet, dans la littérature le score du modèle de langue et celui du modèle de traduction direct sont les deux scores les plus importants par rapport aux autres scores [Koehn et al., 2003] [Och, 2002]. Donc, la contrainte est que les deux poids (ML et MT Direct) de ces deux scores doivent toujours être supérieurs aux autres poids. Cette contrainte est reproduite au niveau des fonctions de reproduction et si un des chromosomes résultant du croisement ou de la mutation ne respecte pas la contrainte, alors nous modifions les valeurs de ces deux poids pour qu'ils soient supérieurs aux autres.

Reproduction

Le fait que nous manipulons des chromosomes encodant des vecteurs de valeurs numériques facilite la définition des fonctions de reproduction. Les stratégies de croisement et de mutation que nous définissons dans GAWO, sont les stratégies classiques dans la littérature des AG [Holland, 1973], [Hüe, 1997]. Dans les deux points suivants, nous expliquons le déroulement pratique des deux fonctions de croisement et de mutation.

Croisement : la fonction de croisement sélectionne de manière aléatoire deux chromosomes c_a et c_b de la population, et sélectionne une position aléatoire s dans les deux chromosomes pour les croiser (voir Figure 4.3). La fonction de croisement produit deux nouveaux chromosomes c_c et c_d, où $c_c = c_a[1:s] + c_b[s+1:8]$ et $c_d = c_b[1:s] + c_a[s+1:8]$. Donc, le premier chromosome enfant (c_c) est composé des poids à gauche de s plus le poids à la position s de c_a combinés avec les poids à droite de s de c_b. Le deuxième chromosome enfant (c_d) est construit de manière inverse par rapport au premier.

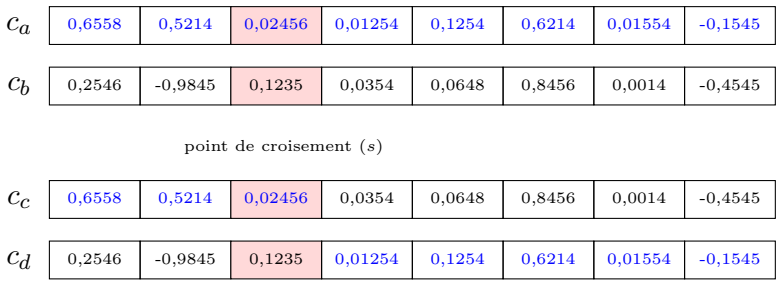


FIGURE 4.3 – Un exemple de croisement dans l'algorithme génétique dans la boucle interne de GAWO.

Nous appliquons la fonction de croisement pour coupler toutes les paires de chromosomes possibles dans la population, et nous produisons pour chaque paire deux nouveaux chromosomes. Avec une population de 500 chromosomes, à chaque itération, 124 750 paires sont croisées pour produire 125 750 2 = 249 500 nouveaux chromosomes. À la fin de chaque itération et au cours du mécanisme de sélection, les chromosomes en double sont éliminés pour éviter une convergence rapide vers un optimum local.

Dans la littérature des AG [Holland, 1973], la fonction de croisement doit coupler tous les chromosomes d'une population pour exploiter idéalement le potentiel de la population. Cela est possible en matière de temps de calcul pour ce problème, vu que nous manipulons des vecteurs de 8 valeurs numériques. Pour d'autres problèmes, comme notre décodeur génétique GAMaT, la complexité de l'encodage du chromosome, nous oblige à limiter le nombre de fois que nous appliquons le croisement à cause des limites techniques et aussi à contraindre l'application de cette fonction sur certaines paires de chromosomes pour éviter de sortir de l'espace de recherche.

Mutation : comme présenté précédemment, la fonction de croisement couple les chromosomes existant dans la population, ce qui limite l'espace de recherche aux valeurs des poids générés à l'initialisation. Par conséquent, l'algorithme a une forte probabilité de converger vers un optimum

local.

Quant à la fonction de mutation, elle est appliquée afin de diversifier la population et tester de nouvelles valeurs des poids qui n'existaient pas auparavant dans la population.

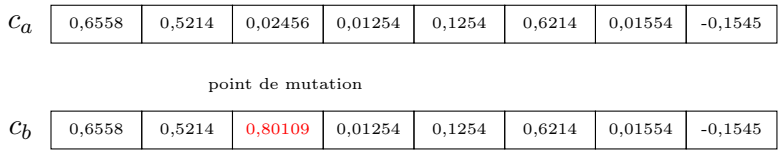


FIGURE 4.4 – Un exemple de mutation dans l'algorithme génétique dans la boucle interne de GAWO.

En pratique, la fonction de mutation sélectionne aléatoirement un chromosome c_a de la population, et sélectionne, par la suite, un des gènes de c_a de manière aléatoire aussi. Une nouvelle valeur est générée pour ce gène de manière aléatoire dans l'intervalle $[-1, 1]$ afin de produire un nouveau chromosome c_b (voir Figure 4.4).

Pour une meilleure diversification de la population et une meilleure exploration de l'espace de recherche, la fonction de mutation est appliquée sur tous les chromosomes de la population. La mutation est appliquée une seule fois sur chaque chromosome de la population et ceux issus des croisements, ce qui nous permet de produire 500 + 249 500 nouveaux chromosomes à chaque itération.

Fonction *fitness*

Deux métriques d'évaluation de la qualité des traductions (BLEU et TER) sont testées pour la définition de la fonction *fitness* des chromosomes. Si nous utilisons le BLEU comme métrique d'évaluation, alors l'évaluation d'un chromosome c_a est faite comme suit :

- en utilisant l'ensemble des poids λ encodés dans le chromosome c_a, nous recalculons les scores de la fonction log-linéaire de toutes les traductions produites par GAMaT dans la boucle externe $E = \{E_1, ..., E_n\}$.
- pour chaque phrase source f_i, nous sélectionnons à partir de son ensemble de k traductions E_i la traduction $\hat{e}_i$ qui maximise le score log-linéaire.
- le résultat de l'étape précédente est un ensemble de n traductions $\hat{e}^{\lambda} = \{\hat{e}_1^{\lambda}, ..., \hat{e}_n^{\lambda}\}$ qui maximisent les scores de la fonction log-linéaire en utilisant les poids λ du chromosome c_a.
- enfin, nous calculons le score BLEU entre les traductions sélectionnées $\hat{e}^{\lambda} = \{\hat{e}_1^{\lambda}, ..., \hat{e}_n^{\lambda}\}$ et les traductions de référence $r = \{r_1, ..., r_n\}$. Le score BLEU obtenu représente le score *fitness* du chromosome c_a.

De cette manière, l'ensemble optimal de poids est celui qui est encodé dans le chromosome qui permet à sélectionner un ensemble de n traduction maximisant le score BLEU. D'un autre côté, si nous utilisons la métrique TER, alors l'ensemble optimal de poids est celui qui est encodé dans le chromosome qui minimise le score TER.

Sélection et remplacement

Dans l'algorithme génétique de GAWO, les fonctions de reproduction sont appliquées sur tous les chromosomes de la population, donc il n'y a pas de notion de sélection qui est définie. Le mécanisme de remplacement des chromosomes à la fin de chaque itération est défini sur la base d'une stratégie de sélection ordinale (voir Section 2.3.5 du Chapitre 2), où les chromosomes les mieux évalués sont gardés pour l'itération suivante.

En appliquant GAWO dans GAMaT pour l'optimisation des poids de la fonction log-linéaire, nous espérons affiner la *fitness* de GAMaT afin d'améliorer ces prises de décisions au niveau des mécanismes de sélection et de remplacement. Dans le chapitre des expérimentations, nous étudions l'impact de GAWO sur les performances de traduction de GAMaT en présentant différents tests et résultats.

La combinaison de ces deux algorithmes génétiques, GAMaT pour le décodage et GAWO pour l'optimisation des poids, nous permet de mettre en place un système de TAS à base de segments ayant un décodeur entièrement basé sur les algorithmes génétiques. Dans le chapitre suivant, nous présentons différentes expérimentations pour évaluer les performances de traductionsde ce système de traduction et pour évaluer ses différents composants.

4.3 Réseau de neurones pour l'évaluation des hypothèses de traduction

Dans la littérature de la TAS, la fonction d'évaluation des hypothèses de traduction au cours du décodage est définie en utilisant l'approche log-linéaire (voir la Section 1.4.4 du Chapitre 1). Nous utilisons aussi cette approche pour définir la fonction *fitness* de l'algorithme génétique de décodage GAMaT. Comme nous l'avons présenté dans la section précédente, nous proposons un algorithme génétique pour l'optimisation des poids des différents scores des différents modèles. La fonction log-linéaire pondérée permet de combiner plusieurs scores et de mettre en place une fonction d'évaluation multicritère, mais qui nécessite obligatoirement une adaptation pour chaque paire de langues et pour chaque corpus de texte utilisé. Cette adaptation est faite à travers le processus d'optimisation des poids des différents scores combinés.

Dans ce travail de thèse, nous proposons une nouvelle fonction pour l'évaluation des chromosomes en utilisant un réseau de neurones [Douib et al., 2017b]. L'idée de la nouvelle fonction est de ne pas utiliser l'approche log-linéaire et de ne pas optimiser ses poids. La fonction apprise par le réseau de neurones prend en entrée les 8 scores et elle prédit la valeur BLEU de la traduction encodée dans un chromosome. En d'autres termes, nous voudrions une fonction *fitness* qui mette en corrélation les 8 scores d'évaluation d'un chromosome et le score BLEU [Papineni et al., 2002] de la traduction encodée dans le chromosome.

Ce type d'algorithmes d'apprentissage est largement utilisé dans la communauté de l'estimation de la qualité [Bojar et al., 2015], afin d'estimer la qualité d'une traduction sans avoir accès à la traduction de référence. À cette fin, les algorithmes d'apprentissage mis en place apprennent une fonction sur des caractéristiques extraites des paires de phrases sources et de leur traductions. Ces caractéristiques sont corrélées à une métrique d'évaluation, pouvant être une métrique à valeurs binaires (bon/mauvais) ou une métrique avec valeurs continues (BLEU, TER, etc.).

Dans la littérature, différents algorithmes d'apprentissage automatique peuvent être utilisés pour l'apprentissage de cette fonction, tels que le SVM ou les réseaux de neurones. Chaque année, une compétition mondiale [Bojar et al., 2017] est organisée pour l'estimation de la qualité des traductions, où les participants proposent de nouveaux algorithmes d'apprentissage ainsi que de nouvelles caractéristiques pour l'apprentissage.

Dans ce qui suit, nous présentons l'architecture du réseau de neurones utilisée pour l'apprentissage de la nouvelle fonction *fitness* et nous décrivons également, comment nous générons l'ensemble de données pour cet apprentissage.

4.3.1 Architecture du réseau de neurones

Comme présenté auparavant, le but de ce travail est de proposer une nouvelle fonction *fitness* pour les chromosomes de GAMaT, qui combine de manière optimale les différents scores présentés précédemment et qui est corrélée avec le score BLEU. Pour ce faire, nous avons décidé d'utiliser un réseau de neurones de régression pour un apprentissage supervisé [Douib et al., 2017b]. Le réseau de neurones prend en entrée les 8 scores d'un chromosome et comme étiquette en sortie, la valeur BLEU de la traduction encodée dans le même chromosome. Ce réseau de neurones est illustré dans la figure 4.5. La fonction *fitness* résultante de cet apprentissage est supposée prédire le bon score BLEU de l'hypothèse de traduction encodée dans un chromosome en fonction de ses 8 scores d'évaluation.

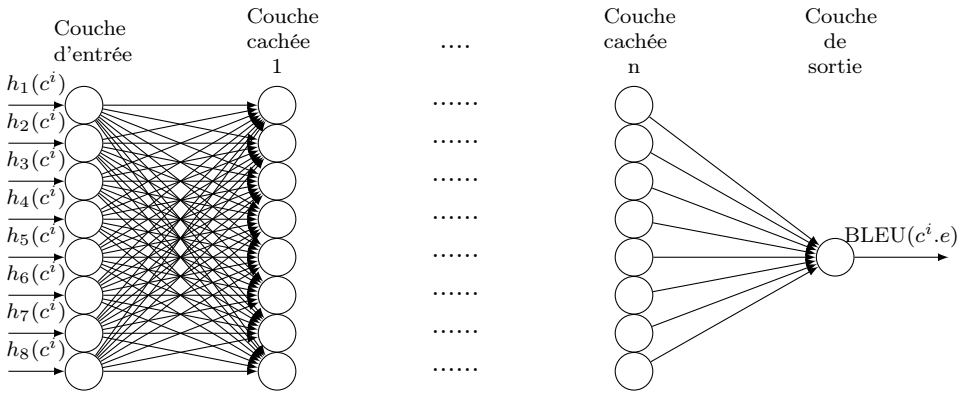


FIGURE 4.5 – L'architecture du réseau de neurones pour la prédiction du score BLEU.

En pratique, nous expérimentons différentes configurations de ce réseau de neurones en faisant varier le nombre de couches cachées et le nombre de neurones des différentes couches cachées. Dans le chapitre suivant, les performances d'apprentissage de ces différentes configurations sont présentées en détail. Pour l'activation des neurones, nous utilisons la fonction Sigmoid pour tous les neurones du réseau.

4.3.2 Génération des données pour l'apprentissage

L'apprentissage que nous proposons à ce niveau est une nouvelle idée, ce qui fait qu'il n'existe pas de corpus d'apprentissage pour apprendre la fonction *fitness*. Du coup, il est nécessaire de

créer les données adéquates et former un corpus d'apprentissage afin de définir une nouvelle fonction *fitness* pour chaque paire de langues traitée.

Le corpus à créer doit être composé d'un nombre important de paires $< v_c, \alpha >$, où v_c est un vecteur de 8 scores à valeurs numériques, qui évaluent différents aspects du chromosome c, que nous avons présentés dans le chapitre précédent. α représente la valeur BLEU de l'hypothèse de traduction encodée dans le chromosome c. Nous utilisons notre décodeur génétique GAMaT pour produire plusieurs chromosomes de différentes qualités. Ces chromosomes sont produits à partir d'un corpus parallèle, contenant des phrases sources et leur traduction de référence.

Pour chaque phrase source du corpus parallèle, nous utilisons les fonctions d'initialisation de GAMaT pour produire plusieurs chromosomes différents. D'autres chromosomes sont produits à travers les fonctions de croisement et de mutation. La différence entre ces chromosomes se situe au niveau de la segmentation des phrases sources, les alignements entre les segments sources et cibles et les traductions des segments sources. Nous devons produire des chromosomes encodant des traductions de différentes qualités (parfaites, bonnes et mauvaises) afin de diversifier les données d'apprentissage. Cette diversification doit permettre à la fonction *fitness* de prédire correctement la qualité de la traduction encodée dans un chromosome quelle que soit cette qualité.

Nous listons dans ce qui suit les différentes stratégies que nous utilisons pour diversifier la qualité des chromosomes pour chaque phrase source :

— Nous utilisons les 5 fonctions d'initialisation de GAMaT (voir Section 3.3 du Chapitre 3) pour générer plusieurs chromosomes de différentes qualités.

— Nous rajoutons une nouvelle fonction d'initialisation qui a un comportement de segmentation aléatoire, mais qui sélectionne pour chaque segment source la traduction ayant la probabilité de traduction la plus faible dans la TT. Cette fonction est utilisée pour produire des chromosomes encodant des traductions de mauvaise qualité.

— Pour chaque phrase source, nous conservons le meilleur chromosome produit en sortie par GAMaT. Cela permet d'avoir pour chaque phrase source un chromosome avec une traduction de bonne qualité.

— De la même manière que la stratégie précédente, nous prenons la meilleure solution proposée par MOSES pour créer à partir de celle-ci un autre chromosome de bonne qualité.

— Les phrases sources pour lesquelles il est possible de produire la traduction de référence en puisant dans la TT, nous créons un chromosome encodant la traduction de référence. Ce chromosome est produit grâce à une option disponible sur MOSES, dans laquelle le décodeur prend une phrase source et sa traduction, de référence dans ce cas, et essaye de trouver les segmentations et alignements permettant de traduire la source vers cette cible tout en maximisant le score du modèle de traduction direct. Cette stratégie permet de produire des chromosomes encodant des traductions parfaites car leur score BLEU vaut 1.

Nous utilisons ces stratégies pour produire un grand nombre de chromosomes de différentes qualités pour toutes les phrases sources du corpus parallèle. Nous obtenons un ensemble $C = \{c_1^1, c_1^2, ..., c_1^n, ..., c_m^1, c_m^2, ..., c_m^n\}$ de $n \times m$ chromosomes, où n est le nombre de chromosomes produits pour chaque phrase source et m est le nombre total de phrases sources dans

le corpus parallèle. Pour chaque chromosome c_i^j de cet ensemble, nous calculons les 8 scores d'évaluation afin de produire l'ensemble des scores $v_i^j = \{h_1, h_2, ..., h_8\}$. Les données en entrée pour le réseau de neurones sont donc $n \times m$ lignes de vecteurs de scores de chromosomes $V = \{v_1^1, v_1^2, ..., v_1^n, ..., v_m^1, v_m^2, ..., v_m^n\}$.

Pour chaque phrase source f_i, nous avons sa traduction parallèle de référence r_i que nous utilisons pour calculer le score BLEU de la traduction de e_i. Donc, pour chaque chromosome c_i^j, nous calculons le score BLEU de la traduction $c_i^j.e$ par rapport à la référence r_i. Le score BLEU de chaque chromosome c_i^j définit sa valeur α_i^j. Les données en sortie du réseau de neurones sont donc $n \times m$ étiquettes $A = \{\alpha_1^1, \alpha_1^2, ..., \alpha_1^n, ..., \alpha_m^1, \alpha_m^2, ..., \alpha_m^n\}$, où chaque valeur de BLEU α_i^j est associée au vecteur v_i^j.

Le résultat final de cette étape de création de données d'apprentissage est un ensemble de données de $n \times m$ lignes :

$$Data = \{< v_1^1, \alpha_1^1 >, ..., < v_1^n, \alpha_1^n >, ..., < v_m^1, \alpha_m^1 >, ..., < v_m^n, \alpha_m^n >\}$$

Pour un réseau de neurones, plus il y a de données plus la qualité de l'apprentissage est meilleure. Pour notre problème, le nombre de données d'apprentissage à produire dépend de la disponibilité des corpus parallèles. Pour certaines paires de langues, comme pour l'anglais et le français, il existe plusieurs corpus composés de millions de phrases parallèles [Bojar et al., 2017], ce qui permet de produire un corpus d'apprentissage riche et diversifié. Cependant, pour d'autres paires de langues, comme pour l'anglais et le turc, les corpus parallèles qui existent ne contiennent que des centaines de milliers de phrases parallèles. Pour ce genre de paires de langues, nous sommes contraints de produire un petit corpus d'apprentissage pour le réseau de neurones ce qui affaiblit les performances d'apprentissage, et par conséquent cela affaiblit la qualité de la fonction *fitness* à définir pour GAMaT.

Dans le chapitre suivant, nous présentons les corpus d'apprentissage que nous avons générés tout en détaillant leur distributions en matière de qualité de traduction. Les expérimentations et les résultats sont également présentés.

4.3.3 Réseau de neurones comme fonction *fitness* de GAMaT

Une fois le réseau de neurones lancé et la fonction $BLEU_{RN}$ de prédiction du score BLEU apprise, nous pouvons l'utiliser comme fonction *fitness* au cours du processus génétique de GAMaT. Cette fonction doit être apprise pour chaque paire de langues à traiter. La fonction d'évaluation des chromosomes est donc définie comme suit :

$$Fitness(c) = BLEU_{RN}(c) \tag{4.3}$$

La fonction $BLEU_{RN}(c)$ prend en entrée les 8 scores qui estiment les différents aspects du chromosome c et prédit le score BLEU de la traduction $c.e$ encodée dans le chromosome c. Donc le meilleur chromosome dans GAMaT est celui qui a le score *fitness* le plus élevé.

Comparée à l'approche log-linéaire, la combinaison des différents scores se fait à travers l'apprentissage du réseau de neurones dans lequel les 8 scores sont corrélés avec le score BLEU de la traduction encodée dans le chromosome. Dans cette approche, il n'est pas nécessaire d'utiliser des poids afin de normaliser la combinaison ni de processus d'optimisation des poids. Cependant, si nous voulons ajouter d'autres scores dans la combinaison, il sera nécessaire de générer à nouveau des données d'apprentissage et de relancer l'apprentissage du réseau de neurones.

4.4 Conclusion et discussion

Dans ce chapitre, nous avons présenté nos travaux de recherche pour la fonction *fitness* d'évaluation des chromosomes de notre décodeur génétique GAMaT. Nous avons proposé deux approches pour définir cette fonction *fitness*.

La première approche est l'approche log-linéaire [Och and Ney, 2003], qui est largement utilisée dans la communauté de TAS pour évaluer les hypothèses de traduction au cours du décodage. Pour cette approche, nous proposons GAWO [Douib et al., 2017a], un algorithme génétique pour l'optimisation des poids des différents scores combinés dans la fonction log-linéaire. Cette optimisation génétique permet d'adapter la fonction d'évaluation des chromosomes au processus génétique de GAMaT et d'améliorer les performances de traduction de notre système, comme nous le présentons dans la section suivante. Cette combinaison des deux algorithmes génétiques, GAMaT pour le décodage et GAWO pour l'optimisation des poids, nous permet de mettre en place un système de traduction automatique statistique ayant un décodeur entièrement basé sur les algorithmes génétiques.

La deuxième approche, consiste à remplacer totalement la fonction log-linéaire pour définir la fonction *fitness* de GAMaT. Nous proposons un réseau de neurones pour apprendre une nouvelle fonction *fitness* qui prédit le score BLEU de l'hypothèse de traduction encodée dans le chromosome [Douib et al., 2017b]. Le réseau de neurones utilisé met en corrélation les 8 scores d'évaluation d'un chromosome avec le score BLEU de l'hypothèse de traduction encodée dans le chromosome. Cette approche peut être considérée comme alternative à l'approche log-linéaire pour la combinaison des différents scores d'un chromosome sans avoir recours à un processus d'optimisation des poids.

Avec ces deux approches, notre système génétique de TAS à base de segments est complètement construit et nous pouvons l'évaluer. Dans le chapitre suivant, nous présentons différentes expérimentations, où nous testons le décodeur GAMaT avec les deux fonctions *fitness* et nous testons également les différentes stratégies des algorithmes génétiques que nous avons proposées. Les performances de traduction de notre décodeur génétique sont comparées aux performances de deux systèmes de référence dans la littérature, à savoir le décodeur MOSES et un système de traduction neuronal. Nous mettons aussi en place des expérimentations détaillées afin de comparer notre système aux systèmes de référence en traduction automatique.

5

Expériences et résultats

5.1 Introduction

Dans ce dernier chapitre, nous présentons différentes expérimentations pour analyser le comportement de notre décodeur génétique GAMaT et pour analyser ses performances de traduction. Nous analysons aussi, le comportement et les performances des deux approches d'optimisation de la fonction *fitness* de GAMaT. Nous comparons également la qualité de notre décodeur par rapport à deux systèmes de traduction de référence dans la littérature, à savoir MOSES [Koehn et al., 2007], le décodeur de référence en TAS, et un système de traduction neuronal implémenté avec OpenNMT [Klein et al., 2017].

Dans un premier temps, nous détaillons les corpus de données que nous utilisons pour l'apprentissage des différents modèles ainsi que pour l'optimisation et les tests. Par la suite, nous présentons les tests que nous avons mis en place pour analyser les différents composants du décodeur génétique et ses différents paramètres. Nous présentons aussi les tests et expérimentations pour fixer les paramètres des deux approches d'optimisation. À la fin, nous présentons différents résultats de traduction comparatifs, et nous analysons les traductions en sortie des trois systèmes pour étudier les points forts et points faibles de chaque approche de traduction et pour étudier également, une possible combinaison des trois systèmes.

5.2 Corpus et données d'apprentissage

Pour réaliser nos expérimentations, nous utilisons les données de la compétition internationale de traduction automatique *"Workshop on Statistical Machine Translation"* [Bojar et al., 2017]. Ces données sont disponibles gratuitement pour un téléchargement sur le site officiel [3] de la conférence. Plusieurs corpus parallèles, de différentes paires de langues, sont disponibles, et chaque participant les utilise pour créer son propre système de traduction. Les différents systèmes varient en matière d'approche de traduction, d'algorithme d'apprentissage ou d'algorithme de décodage. La compétition consiste à comparer les performances de traduction des différents systèmes proposés.

Dans ce travail de thèse, nous choisissons deux différentes paires de langues, à savoir la paire français et anglais (FR-AN) et la paire turc et anglais (TR-AN), pour réaliser nos différentes

3. http ://www.statmt.org/wmt17/

expérimentations. Le choix de ces deux paires est fait sur la base de la différence linguistique qui peut exister entre les deux paires. En effet, le français et l'anglais sont deux langues proches et pour lesquelles la tâche de traduction est plus simple comparée à la traduction du turc vers l'anglais, où ces deux langues ont une plus forte différence linguistique ce qui rend le processus de traduction plus compliqué. Plus il y a de différence linguistique entre deux langues plus il est difficile d'apprendre les bonnes relations (paires de segments) entre elles à travers les modèles de traduction. D'un autre côté, la taille des corpus a une grande importance dans la qualité des modèles à apprendre. Avec un plus grand nombre de phrases parallèles, les processus d'apprentissage arrivent à détecter plus de motifs de traduction et avec une plus grande fréquence de répétition.

Dans la Tableau 5.1, nous donnons quelques chiffres détaillant les corpus parallèles que nous utilisons pour l'apprentissage des modèles de traductions pour les deux paires de langues FR-AN et TR-AN.

Corpus	**Langue**	**# phrases parallèles**	**Taille vocabulaire**	**Taille moyenne des phrases**
FR-AN	**FR**	1.5M	300K	26.25
	AN	1.5M	240K	25.02
TR-AN	**TR**	200K	150K	22.20
	AN	200K	70K	24.69

TABLE 5.1 – Statiques des corpus d'apprentissage pour les modèles de traduction.

Nous pouvons voir que pour la paire FR-AN, nous avons un bon nombre de phrases parallèles ce qui va nous permettre de mettre en place de bons modèles de traduction et produire par la suite des traductions d'une qualité acceptable comme nous allons le présenter dans la suite de ce chapitre. D'un autre côté, comme la Turquie ne fait pas partie de l'union européenne et que la langue turque est peu utilisée dans les organismes internationaux, nous n'avons que peu de phrases parallèles disponibles pour la compétition. En plus du peu de données pour la paire TR-AN, le fait que la langue turque soit une langue agglutinante, dans laquelle toute une suite de mots (expression) peut être représentée par un seul mot qui concatène plusieurs morphèmes (ex : TR : evlerimdekiler - AN : those who are in my homes). Ce genre de langues complique le processus d'apprentissage du modèle de traduction.

Nous pouvons remarquer aussi, que pour la paire FR-AN, la taille du vocabulaire de l'anglais est plus faible par rapport au français. Cet écart est plus important entre l'anglais et le turc, où pour les 200K phrases parallèles, nous avons 150K mots différents pour la langue turque, contre 70K mots différents pour la langue anglaise, ce qui représente plus que le double. Afin d'apprendre ces modèles de traduction et générer les tables de traduction pour les deux paires, nous utilisons GIZA++ [Och and Ney, 2003], l'outil de référence pour la génération des modèles de traduction et que nous avons présenté dans la Section 1.4.2 du Chapitre 1.

Pour l'anglais, la langue cible des deux paires de langues que nous utilisons pour nos expérimentations, nous utilisons plus de données pour mettre en place un modèle de langue plus riche et permettant une meilleure évaluation des hypothèses de traduction. Ces données sont disponibles aussi dans le cadre de la compétition de traduction automatique [Bojar et al., 2017]. Dans la tableau 5.2, nous détaillons en chiffres le corpus que nous utilisons pour apprendre un modèle de langue 5-grammes à l'aide l'outil SRILM [Stolcke et al., 2011].

Corpus	# phrases	Taille vocabulaire	Taille moyenne des phrases
EN	5.2M	380K	23.71

TABLE 5.2 – Statistiques du corpus cible d'apprentissage pour le modèle de langue.

5.3 Paramétrage du décodeur génétique GAMaT

Dans cette section, nous présentons différentes expérimentations réalisées afin d'étudier le comportement du décodeur génétique GAMaT et afin d'analyser l'influence de la variation de ses paramètres sur son comportement et ses performances de traduction.

5.3.1 Taille de la population et taux de croisement et de mutation

Dans le premier test, nous étudions l'influence de la taille de la population, le taux de croisement et le taux de mutation sur le comportement du processus génétique, en matière de nombre d'itérations nécessaires pour converger, et aussi en matière de qualité des traductions produites sur la base du score BLEU. Nous faisons varier la taille de la population entre 10, 50, 100, 150 et 200 chromosomes. Pour chaque taille de la population, trois chromosomes, de la population initiale, sont générés par les trois premières fonctions d'initialisation favorisant les longs segments et les autres chromosomes sont générés par les deux fonctions à comportement aléatoire. Quant aux taux de croisement et de mutation, nous les faisons varier de 10% à 90% chacun avec des pas de 20%. Dans la littérature [Holland, 1973] le taux de croisement est toujours supérieur au taux de mutation, mais dans ce test nous les faisons varier de la même manière et nous analysons les résultats.

Nous faisons varier ces trois paramètres dans trois boucles imbriquées, et pour chaque configuration de ces trois paramètres, nous lançons GAMaT pour traduire un ensemble de développement de 1 000 phrases sources du français vers l'anglais. À ce niveau, la fonction *fitness* de GAMaT est définie en utilisant l'approche log-linéaire combinant les 8 scores présentés dans la Section 3.6 du Chapitre 3. Les poids associés aux différents scores sont issus d'une optimisation faite par MERT [Och, 2003] et appliquée sur MOSES, que nous utilisons pour toutes ces configurations. Nous utilisons ces mêmes poids pour les prochaines expérimentations dans notre analyse du comportement génétique de GAMaT. Par la suite, nous présentons les performances de GAMaT avec de nouvelles valeurs des poids optimisées par GAWO.

Dans la figure 5.1 et la figure 5.2, nous présentons deux graphiques qui illustrent l'influence de ces trois paramètres sur, respectivement, la qualité des traductions en BLEU et le nombre d'itérations qu'il faut pour converger. Comme nous l'avons présenté dans la Section 3.8 du Chapitre 3, nous utilisons deux critères de convergence du processus génétique de GAMaT, à savoir un nombre maximal d'itérations qui est fixé à 1 000 et qui n'est jamais atteint dans nos expérimentations, et une stagnation de la qualité de la population des chromosomes durant 10 itérations successives. Dans les deux graphiques, nous avons trois axes qui représentent les trois paramètres, et les cercles représentent les scores BLEU (Figure 5.1) et le nombre d'itérations moyen, sur les 1 000 phrases, pour converger (Figure 5.2). La taille d'un cercle illustre sa valeur, et plus un cercle est grand plus le score BLEU lié à la configuration est grand.

Nous pouvons voir qu'il n'y a quasiment pas d'influence de la variation de ces trois paramètres sur le score BLEU des traductions en sortie. Cela se voit à travers la taille des cercles sur

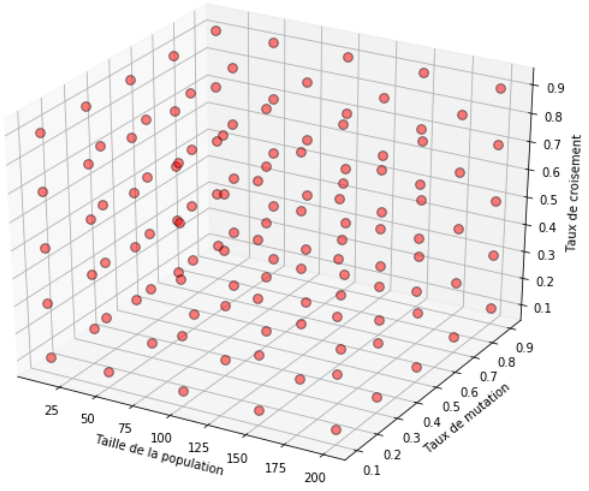


FIGURE 5.1 – Influence sur le score BLEU.

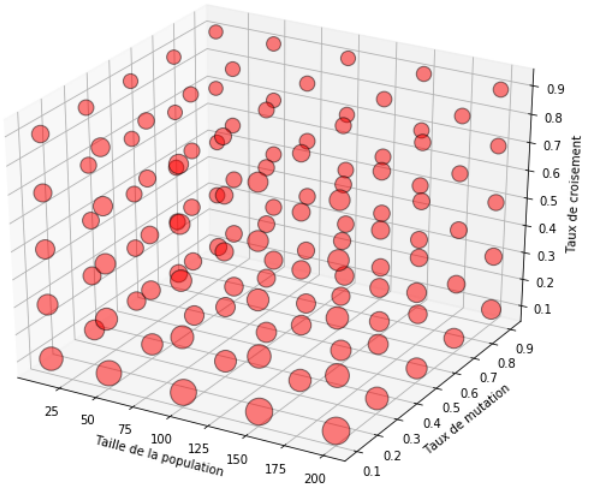


FIGURE 5.2 – Influence sur le nombre d'itérations.

la figure 5.1, qui ont des tailles qui se rapprochent fortement, avec un score moyen de 29.55 en BLEU. D'un autre côté, le nombre d'itérations moyen pour converger est clairement influencé par la variation des trois paramètres. Nous pouvons remarquer, sur la figure 5.2, qu'avec des taux de mutation et de croisement faibles (10%), GAMaT a besoin de plus d'itérations pour converger vers des populations optimales et inversement pour des taux plus élevés (90%). Le plus grand nombre d'itérations (260 itérations), pour converger, est atteint par une population de 200 chromosomes et des taux de mutation et de croisement de 10%. Quant au nombre d'itérations le plus faible (64 itérations), il est obtenu avec une population de 10 chromosomes et des taux de mutation et de croisement de 90%.

Dans la figure 5.3 et la figure 5.4, nous illustrons l'évolution décroissante du nombre d'itéra-

tions nécessaires pour la convergence en fonction des taux de croisement et de mutation tout en fixant la taille de la population à 100 chromosomes.

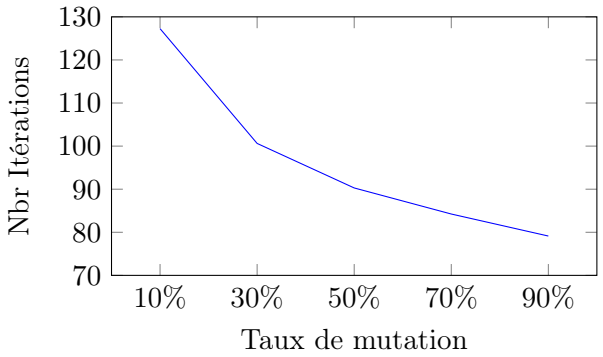


FIGURE 5.3 – Évolution du nombre d'itérations en fonction du taux de mutation. Taille de la population fixée à 100 et taux de croisement fixé à 70%.

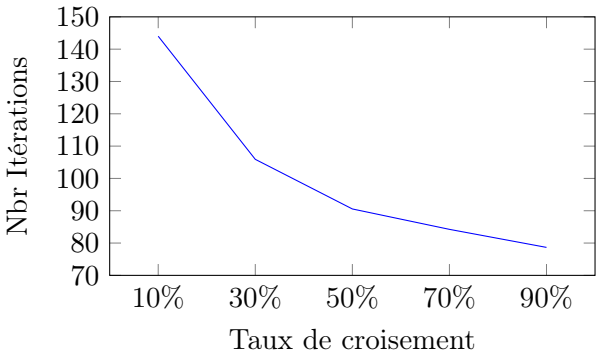


FIGURE 5.4 – Évolution du nombre d'itérations en fonction du taux de croisement. Taille de la population fixée à 100 et taux de mutation fixé à 70%.

Ce que nous pouvons conclure, à partir de ces premiers résultats, c'est que ces trois paramètres n'ont pas d'influence sur les performances de traduction de GAMaT, où toutes les configurations permettent à GAMaT de converger et de produire des traductions ayant un score BLEU de 29.55 en moyenne. La différence entre ces configurations est au niveau du nombre d'itérations qu'il faut pour converger. Pour la suite des tests et expérimentations, nous fixons la taille de la population à 100 chromosomes et les taux de croisement et de mutation à 70%. Nous optons pour cette configuration, car elle permet, d'une part, une convergence rapide (84 itérations en moyenne), et d'autre part, elle a permis d'avoir un score de 29.68 en BLEU, un des scores les plus élevés de toutes les configurations testées.

5.3.2 Politique de sélection et de remplacement

Dans les expérimentations précédentes, nous avons utilisé une sélection ordinale pour définir les deux mécanismes de sélection et de remplacement. Dans cette partie, nous comparons les deux stratégies, à savoir une stratégie de sélection ordinale et une sélection à base de roulette, comme présenté dans la Section 3.7 du Chapitre 3. Afin de faire cette comparaison, nous lançons GAMaT pour traduire un ensemble de développement de 1 000 phrases sources du français vers l'anglais. Le premier scénario est à base d'une sélection ordinale et le deuxième est à base d'une sélection à base de roulette. Pour les deux scénarios, la taille de la population et les taux de croisement et de mutation sont fixés aux valeurs présentées dans la section précédente.

Stratégie	**BLEU**	**Nombre d'itérations moyen**
Ordinale	29.59	86
Roulette	29.68	98

TABLE 5.3 – Choix de stratégie de sélection et de remplacement pour GAMaT.

À travers les résultats que nous présentons dans la Tableau 5.3, nous pouvons remarquer que

la stratégie de sélection à base de roulette permet à GAMaT de produire des traductions de meilleure qualité en matière de score BLEU comparée à l'utilisation de sélection ordinale, mais cette différence, de 0,09 de BLEU, est faible et non significative sur un ensemble de 1 000 phrases. Ces performances montrent aussi que GAMaT arrive à converger pour les deux stratégies et la différence est au niveau du nombre d'itérations nécessaires pour converger, où la stratégie à base de roulette nécessite plus d'itérations pour converger.

Sur la base de ces résultats, pour la suite des expérimentations, nous utilisons une sélection à base de roulette pour les deux mécanismes de sélection et de remplacement. Cette stratégie permet à GAMaT d'avoir de meilleures performances même si la différence est faible par rapport à la stratégie ordinale.

5.3.3 Convergence dans GAMaT

Dans un déroulement correct d'un processus génétique, la population de chromosomes doit s'améliorer d'une génération à une autre. Dans les premières générations, cette évolution peut être de manière irrégulière, mais le comportement global du processus, à travers les différentes générations, doit évoluer vers des populations meilleures que les populations initiales.

Afin de vérifier le comportement évolutif de notre décodeur génétique GAMaT, nous prenons quatre phrases sources en français, extraites de l'ensemble de développement, et nous les traduisons vers l'anglais. Pour chaque phrase source, nous évaluons l'évolution du score de la *fitness* du meilleur chromosome à chaque itération et aussi l'évolution du score moyen des *fitness* des chromosomes de chaque itération. Nous listons dans ce qui suit ces quatre phrase sources :

— *"Ils ont à leur disposition différents moyens pour le faire, conformément aux traditions constitutionnelles."* (voir Figure 5.5). Cette phrase est composée de 14 mots.

— *"Pour toute puissance et, en particulier, pour une superpuissance éminemment civiles comme l'union, la politique étrangère est un mélange de questions commerciales, économiques, environnementales, de développement, de sécurité, de droits de l'homme."* (voir Figure 5.6). Cette phrase est composée de 34 mots.

— *"La mise en œuvre urgente de ces mesures entraîne la création d'un cadre institutionnel solide et cohérent comme l'a déjà évoqué le parlement et le rappelle ce rapport. Or cette question a recueilli peu d'attention de la part du conseil."* (voir Figure 5.7). Cette phrase est composée de 43 mots.

— *"Je dois souhaiter le succès à Mme Anita Gradin - car son travail au sein de l'UE consiste bien à tenter de venir à bout du problème, même si les choses avancent lentement."* (voir Figure 5.8). Cette phrase est composée de 33 mots.

À travers ces quatre exemples, nous pouvons remarquer que l'évolution, dans le processus de décodage génétique de GAMaT, se déroule correctement. Le score *fitness* du meilleur chromosome et le score *fitness* moyen de la population évoluent de manière croissante du début du traitement jusqu'à une convergence et une stagnation vers des populations optimales. En effet, pour les quatre exemples de traductions, le meilleur chromosome final est obtenu plus rapidement que la convergence de la population des chromosomes. Ce meilleur chromosome peut être produit dès l'initialisation ou au cours des premières générations comme nous pouvons le voir dans la figure 5.5. Dans ce premier exemple, la solution optimale est trouvée dès la première génération après

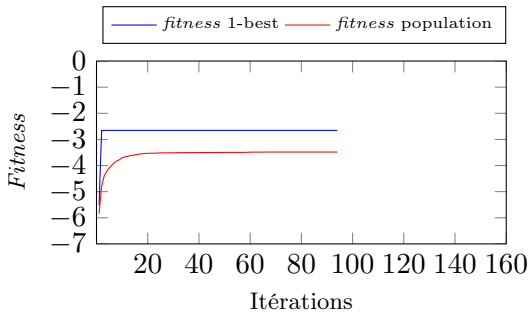


FIGURE 5.5 – Évolution du score *fitness* pour la phrase 1

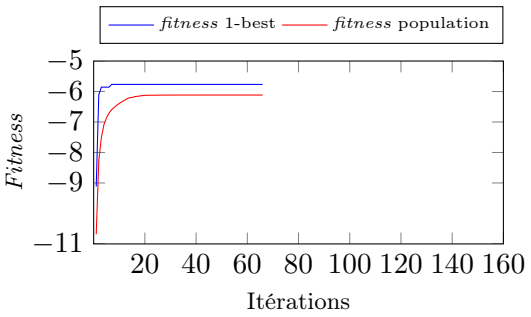


FIGURE 5.6 – Évolution du score *fitness* pour la phrase 2

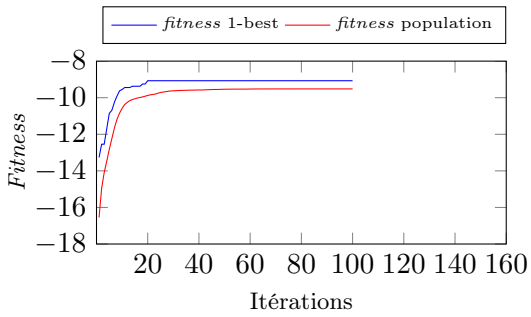


FIGURE 5.7 – Évolution du score *fitness* pour la phrase 3

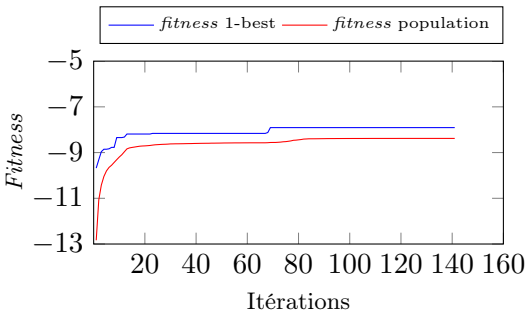


FIGURE 5.8 – Évolution du score *fitness* pour la phrase 4

l'initialisation avec un score *fitness* de -2,655. Quant à la population, elle évolue plus lentement, allant d'un score *fitness* moyen de -4,79, à la première itération après initialisation, jusqu'à un score moyen de -3,48, à la $73^{ème}$ itération. Pour le $4^{ème}$ exemple (voir Figure 5.8), le meilleur chromosome évolue et s'améliore sur plusieurs générations et la solution optimale est produite à la $69^{ème}$ génération après différentes transformations génétiques (croisements et mutations). En parallèle la population des chromosomes évolue de manière remarquable, où le score *fitness* moyen est de -12.83 à la première itération et évolue vers un score moyen de -8.38 au bout de 120 générations.

Nous pouvons remarquer que le nombre de générations nécessaires pour la convergence varie d'un exemple à un autre et cela en fonction de la structure de la phrase source et par conséquent en fonction de son espace de recherche. En effet, l'espace de recherche de la première phrase est plus petit que les autres, vu que cette phrase n'est composée que de 14 mots source, cela peut expliquer la convergence rapide du décodage. Pour cet exemple les fonctions d'initialisation de GAMaT arrivent à produire des gènes de bonne qualité permettant une convergence rapide. La longueur de la phrase source et la taille de son espace de recherche ne sont pas les seuls paramètres de convergence. Nous pouvons voir que pour la deuxième phrase (voir Figure 5.6) la convergence est rapide, même si elle a la même taille que la quatrième (voir Figure 5.8). Pour la troisième phrase (voir Figure 5.7), qui est la plus longue de ces exemples avec 43 mots, le meilleur chromosome est produit à la $20^{ème}$ itération. La structure et la complexité de la phrase source sont un autre paramètre qui a une influence sur la convergence. En effet, dans la deuxième phrase, nous pouvons remarquer une formulation simple et l'existence de plusieurs virgules, cela

facilite le processus de segmentation pour trouver les bonnes frontières entre segments, facilitant par la suite le processus de recherche pour une convergence rapide. Quant à la quatrième phrase, sa structure est plus complexe par rapport aux autres, ce qui pousse le décodeur à prendre plus d'itérations pour converger.

Génération	**Meilleure hypothèse**
1	i must wish success in mrs anita gradin - because its working within the eu are to try to tackle it , even if the situation are slowly .
2	i must wish success in mrs anita gradin - because its working within the eu is to try to overcome the problem , although things are moving slowly .
3	i must wish success in mrs anita gradin - because his work in the eu is to try to overcome the problem , although things are moving slowly .
4	i must wish success in mrs anita gradin - because his work in the eu is to try to come to the problem , even if the meantime slowly .
5	i must wish success in mrs anita gradin - because his work in the eu is to try to come to the problem , even if the meantime slowly .
6	i must wish success in mrs anita gradin - because his work in the eu is of trying to to overcome the problem , even though things are moving slowly .
7	i must wish success in mrs anita gradin - because its working within the eu are to try to tackle it , even if the meantime slowly .
8	i must wish success in mrs anita gradin - because its working within the eu are to try to tackle it , even if the meantime slowly .
9	i must wish success in mrs anita gradin - because his work in the eu are to try to tackle it , even if the meantime slowly .
10	i must wish success in mrs anita gradin - because his work in the eu are to try to tackle it , even if the meantime slowly .
11	i must wish success in mrs anita gradin - because his work in the eu are to try to tackle it , even if the meantime slowly .
12	i must wish success in mrs anita gradin - because his work in the eu are to try to come to end of the problem , although things are moving slowly .
13	i must wish success in mrs anita gradin - because his work in the eu are to try to come to end of the problem , even though things are moving slowly .
...	....
22	i must wish success in mrs anita gradin - because his work in the eu are to try to come to end of the problem , even though things are moving slowly .
23	i must wish success in mrs anita gradin - because his work in the eu are to try to come to eliminate the problem , even if the meantime slowly .
...	....
67	i must wish success in mrs anita gradin - because his work in the eu are to try to come to eliminate the problem , even if the meantime slowly .
68	i must wish success in mrs anita gradin - because his work in of the eu are to try to to overcome the problem , even though things are moving slowly .
69	i must wish success in mrs anita gradin - because his work in the eu are to try to to overcome the problem , even though things are moving slowly .

TABLE 5.4 – Évolution de l'hypothèse de traduction encodée dans le meilleur chromosome à chaque itération, de la phrase source *"Je dois souhaiter le succès à Mme Anita Gradin - car son travail au sein de l'UE consiste bien à tenter de venir à bout du problème, même si les choses avancent lentement."* et qui a comme traduction de référence *"I wish Anita Gradin good luck - her work in the EU is now concerned with managing this problem and it will be no easy task."*.

Dans la tableau 5.4, nous illustrons l'évolution des traductions, en présentant l'hypothèse de traduction encodée dans le chromosome ayant le meilleur score *fitness* à chaque itération. Ces hypothèses de traduction, en anglais, sont celles de la phrase source *"je dois souhaiter le succès*

à mme anita gradin - car son travail au sein de l' ue consiste bien à tenter de venir à bout du problème , même si les choses avancent lentement ." en français. Les changements, dans les hypothèses de traduction, d'une itération à une autre, sont marqués en bleu sur les différentes lignes du Tableau 5.4. L'évolution que nous présentons ne représente pas l'évolution d'un même et seul chromosome tout au long du processus génétique de GAMaT, mais juste de l'hypothèse encodée dans le meilleur chromosome à la fin de chaque itération.

Pour analyser l'évolution de la qualité des hypothèses de traduction, de l'initialisation à la convergence du processus de décodage de GAMaT en fonction du score BLEU, nous reprenons le même ensemble de développement de 1 000 phrases sources en français et nous créons trois ensembles de traductions (voir la tableau 5.5). Le premier ensemble est celui des 1 000 hypothèses de traduction, des phrases sources, encodées dans les meilleurs chromosomes à l'initialisation. Le deuxième est celui des 1 000 hypothèses de traduction encodées dans les meilleurs chromosomes au début de la deuxième itération. Le troisième et dernier ensemble est celui des 1 000 traductions en sortie du décodeur GAMaT. Pour chaque ensemble, nous calculons le score BLEU par rapport aux traductions de référence, en anglais, afin d'estimer la qualité des traductions de l'ensemble. De cette manière, nous pouvons voir la différence de la qualité des traductions au début du processus génétique et en sortie.

Génération	**BLEU**
Initialisation	25,72
Deuxième itération	28,08
Sortie de GAMaT	29,60

TABLE 5.5 – Évolution du score BLEU de l'initialisation à la sortie de GAMaT.

Nous remarquons, à travers les performances du Tableau 5.5, qu'il y a une nette amélioration de la qualité des traductions, sur l'ensemble de développement, de l'initialisation à la sortie du décodeur. En effet, le score BLEU à l'initialisation est de 25,72 et s'améliore de 2,36 points, dès la fin de la première itération de traitement, pour arriver à un score BLEU de 28.08. En sortie du système, le score BLEU atteint les 29,60 marquant une amélioration de 1,52 points par rapport à la première itération et une amélioration de 3,88 points par rapport aux hypothèses de traduction à l'initialisation. Ces résultats montrent que notre décodeur génétique arrive à améliorer la qualité des hypothèses de traduction, des populations initiales, et arrive à converger vers de meilleures populations de chromosomes permettant de proposer en sortie de bonnes traductions. Dans les sections suivantes, nous présentons des tests comparatifs pour situer les performances de notre système avec ceux de l'état de l'art sur des ensembles de test.

5.4 Performances de GAMaT optimisé par GAWO

Dans la section précédente, nous avons présenté des expérimentations sur le comportement génétique du décodeur génétique GAMaT. Sur la base des ces expérimentations, nous avons fixé les différents paramètres de l'algorithme génétique de GAMaT et nous nous sommes assurés de la bonne évolution du processus génétique de décodage.

La fonction *fitness*, que nous avons utilisée, pour l'instant, pour évaluer les chromosomes dans GAMaT, est définie sur la base de l'approche log-linéaire combinant 8 scores différents, que nous

avons présentés dans la Section 3.6 du Chapitre 3. Les poids associés aux différents scores que nous avons utilisés pour définir la fonction *fitness* sont issus d'une optimisation faite par MERT [Och, 2003] (voir Section 1.4.5 du Chapitre 1) appliquée à MOSES. Cette optimisation des poids de la fonction log-linéaire, a été faite pour les deux paires de langues que nous avons présentées au début de ce chapitre, à savoir le français-anglais (FR-AN) et turc-anglais (TR-AN). Pour les deux paires de langues, les ensembles d'optimisation sont composés de 1 000 phrases parallèles.

Dans la tableau 5.6, nous présentons les performances de traduction de notre décodeur génétique GAMaT avec la fonction log-linéaire comme fonction *fitness* et les poids issus de l'optimisation sur MOSES. Nous traduisons deux ensembles de test de 1 000 phrases sources pour les deux paires de langues (FR-AN et TR-AN) et nous évaluons la qualité des traductions en sortie. L'évaluation est faite en utilisant le BLEU et le TER, les deux métriques d'évaluation des traductions, présentées dans la Section 1.5.2 du Chapitre 1. Nous comparons les performances de GAMaT avec celles de MOSES pour les deux paires sur le même ensemble de test pour chaque paire.

Paire de langues	**Système**	**BLEU ↑**	**TER ↓**
FR-AN	**MOSES**	31,29	52,14
	GAMaT	24,84	57,18
TR-AN	**MOSES**	10,29	83,03
	GAMaT	6,46	82,05

TABLE 5.6 – Performances primaires de GAMaT, comparées à celles de MOSES.

Ces premières performances de traduction montrent que notre décodeur génétique GAMaT est nettement moins performant que MOSES, pour les deux paires de langues. En effet, en BLEU, nous constatons 6,45 points de différence entre les deux systèmes pour la paire FR-AN et 3,83 points de différence pour la paire TR-AN. En TER, MOSES dépasse GAMaT par 5,04 points pour la paire FR-AN. Pour la paire TR-AN, GAMaT arrive à faire mieux que MOSES et le dépasse de 1 point.

Cet écart de qualité de traduction entre les deux systèmes peut s'expliquer par le fait que les poids associés aux 8 scores sont optimisés sur le processus de décodage de MOSES et non pas celui de GAMaT. Dans ce cas, la fonction *fitness* de GAMaT n'est pas adaptée à son processus de décodage génétique.

Comme nous l'avons présenté dans la Section 4.2 du chapitre précédent, GAWO est l'algorithme génétique que nous proposons pour optimiser ces poids sur GAMaT afin d'adapter au mieux sa fonction *fitness* et afin d'améliorer ses mécanismes de sélection et de remplacement. Dans les points qui vont suivre, nous présentons des expérimentations de l'utilisation de GAWO appliqué à GAMaT. En premier, nous détaillons les paramètres de l'algorithme génétique de GAWO. Par la suite, nous présentons différentes expérimentations pour analyser le processus d'optimisation des poids de la fonction log-linéaire appliquée à GAMaT. Nous terminons en présentant les performances de traduction de GAMaT avec les nouvelles valeurs des poids.

5.4.1 Paramètres de GAWO

Comme nous l'avons présenté auparavant, le processus d'optimisation de GAWO se déroule en deux boucles imbriquées, à savoir la boucle externe pour la traduction de l'ensemble de développement et la production de k traductions pour chaque phrase source, et la boucle interne, dans laquelle un algorithme génétique pour l'optimisation des poids est implémenté.

À l'initialisation de la boucle externe, nous générons de manière aléatoire un ensemble de valeurs de poids avec la contrainte suivante : les deux poids associés au score du modèle de langue et celui du score du modèle de traduction direct doivent être toujours supérieurs aux autres vu l'importance de ces deux scores dans un système de traduction automatique statistique. Dans les itérations suivantes, les valeurs des poids issus de l'optimisation, au niveau de la boucle interne, sont utilisées pour relancer GAMaT et traduire l'ensemble de développement à nouveau. Le processus itératif de la boucle externe est arrêté dès qu'il arrive à une stagnation, où nous ne pouvons plus produire de meilleures traductions pour l'ensemble de développement. La qualité des traductions, au niveau du corpus, est estimée en utilisant le score BLEU.

Nous avons fixé les différents paramètres de l'algorithme génétique, de la boucle interne, de manière empirique. Comme cet algorithme d'optimisation traite des vecteurs de valeurs numériques, nous avons alors, maximisé les valeurs des paramètres afin de permettre à l'algorithme permettre d'explorer au mieux l'espace de recherche. Nous listons dans ce qui suit ces paramètres et leur valeur.

— **Taille de la population :** nous avons fixé la taille de la population à 500 chromosomes (vecteurs de 8 valeurs numériques). Les 500 chromosomes sont générés de manière aléatoire en respectant la contrainte détaillée précédemment. Nous rajoutons à ces chromosomes le chromosome qui encode le vecteur des valeurs des poids utilisées dans la boucle externe pour la traduction. Nous traitons une population de grande taille afin d'explorer un plus grand nombre de régions de l'espace de recherche et d'évaluer plusieurs solutions.

— **Taux de croisement :** nous faisons croiser toutes les paires de chromosomes possibles de la population afin d'exploiter au mieux la population et afin de produire le maximum de nouvelles solutions.

— **Taux de mutation :** nous faisons muter tous les chromosomes de la population, ainsi que les chromosomes issus des croisements. Comme pour le croisement, la mutation produit un grand nombre de nouveaux chromosomes et facilite l'exploration de nouvelles régions de l'espace de recherche.

— **Sélection :** comme les fonctions de reproduction sont appliquées à la totalité des chromosomes de la population, alors, aucune sélection n'est appliquée dans cet algorithme génétique.

— **Remplacement :** nous utilisons une stratégie de sélection ordinale pour sélectionner les chromosomes à garder d'une itération à une autre.

5.4.2 Analyse du processus d'optimisation de GAWO

Afin d'analyser le comportement de GAWO, nous lançons le processus d'optimisation des poids de la fonction log-linéaire sur GAMaT pour les deux paires de langues (FR-AN et TR-AN) et nous évaluons l'évolution de la qualité des traductions produites par GAMaT durant les ité-

rations successives de la boucle externe.

Dans la figure 5.9 et la figure 5.10, nous traçons deux courbes pour évaluer le comportement de GAWO pour les deux paires de langues. Dans la première courbe (*"boucle interne"*), chaque point représente la valeur BLEU de l'ensemble des meilleures traductions $\{\hat{e}_1, ..., \hat{e}_n\}$ sélectionnées parmi les ensembles des k traductions, en utilisant les meilleures valeurs des poids produites par la boucle interne. En d'autres termes, chaque point représente le score *fitness* du meilleur chromosome (solution optimale) produit par l'algorithme génétique de GAWO dans la boucle interne. La deuxième courbe (*"boucle extérieure"*) représente l'évolution du score BLEU moyen des k traductions, des différentes phrases sources, en sortie du décodage de GAMaT dans la boucle extérieure. Cette dernière courbe, nous permet d'analyser l'évolution de la qualité des populations finales, de traductions produites par GAMaT. Le nombre de traductions produites par GAMaT, pour chaque phrase source, est fixé à 100 (k=100), ce qui correspond à la taille de la population dans GAMaT. Pour la première itération de la boucle externe, nous injectons les 100 x 1000 traductions au processus d'optimisation génétique dans la boucle interne. Par la suite, à chaque itération i, nous rajoutons aux traductions produites par GAMaT les traductions des l'itération $i-1$ afin de diversifier les traductions de chaque phrase source.

Les algorithmes génétiques que nous utilisons au niveau des deux boucles, externe et interne, sont de nature aléatoire, nous avons donc analysé la robustesse et la stabilité de notre algorithme d'optimisation. Nous nous sommes également assurés que GAWO permet toujours à GAMaT de bien converger. Pour vérifier cela, nous lançons le processus d'optimisation 5 fois (voir la figure 5.9 et la figure 5.10).

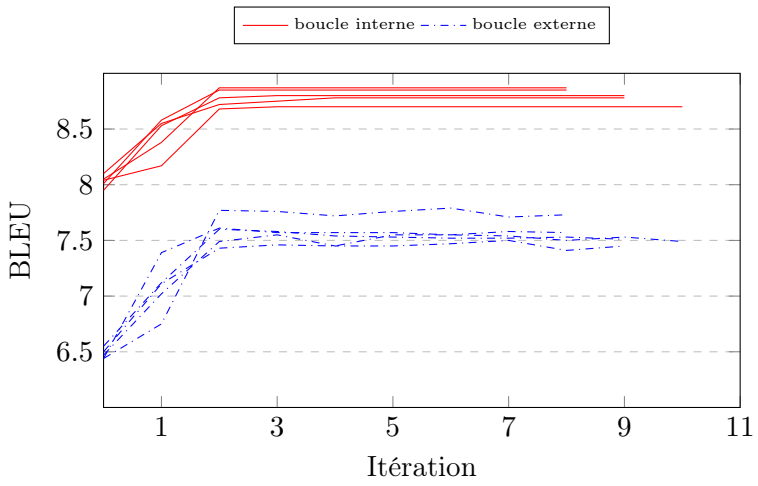


FIGURE 5.9 – L'évolution du score BLEU sur l'ensemble de développement pour la paire TR-AN. La *fitness* de la meilleure solution en sortie de la boucle interne et le score BLEU des 100 traductions en sortie de la boucle externe (GAMaT).

La première remarque que nous pouvons faire, à travers ces deux figures, est que le processus d'optimisation arrive à converger correctement pour les deux paires de langues, et aussi pour les différentes exécutions. Grâce à cette convergence, nous pouvons conclure que GAWO permet à GAMaT d'améliorer ses performances de traduction sur l'ensemble de développement, et cela en générant de meilleures valeurs de poids. Ces valeurs permettent de mieux adapter la fonction *fitness* de GAMaT, ce qui permet d'améliorer ses prises de décision.

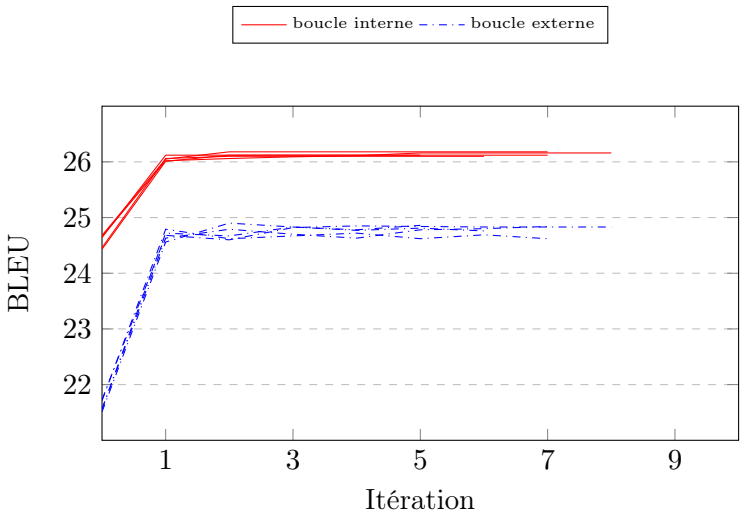


FIGURE 5.10 – L'évolution du score BLEU sur l'ensemble de développement pour la paire FR-AN. La *fitness* de la meilleure solution en sortie de la boucle interne et le score BLEU des 100 traductions en sortie de la boucle externe (GAMaT).

Nous constatons également, qu'en moyenne, trois itérations suffisent pour que le processus d'optimisation atteigne son score maximal. Cela peut s'expliquer par la richesse et la diversité des traductions possibles, pour chaque phrase source, mise à disposition de l'algorithme génétique d'optimisation. En effet, après trois itérations de la boucle externe, nous injectons, dans la boucle interne de GAWO, un grand nombre de traductions pour chaque phrase source ($3 \times k$=300), cela permet de prendre de meilleures décisions et de trouver de meilleures valeurs de poids.

En matière de performances de traduction, dans la boucle interne (voir les courbes rouges dans la figure 5.9 et la figure 5.10), le processus d'optimisation permet d'obtenir un score BLEU moyen de 8,8 pour la paire TR-AN. Pour la paire FR-AN, le score BLEU moyen est de 26,11. Pour la boucle externe (voir les courbes bleues), pour la paire TR-AN, GAMaT produit des populations de traductions, pour l'ensemble des phrases sources, avec un score BLEU moyen autour de 7,6 et un score BLEU moyen autour de 24,83 pour la paire FR-AN. Le score BLEU est plus élevé dans la boucle interne, par rapport à la boucle externe, car dans cette dernière, le score BLEU présenté est la moyenne des scores BLEU des ensembles de k traductions, alors que le score de la boucle interne est celui d'un seul ensemble de n traductions qui maximisent le score BLEU dans l'algorithme génétique d'optimisation.

5.4.3 Performances de traduction de GAMaT optimisé par GAWO

Dans la tableau 5.7, nous présentons les performances de traduction sur l'ensemble de test de 1 000 phrases, en fonction des deux métriques d'évaluation BLEU et TER. Afin d'estimer l'amélioration apportée par l'optimisation sur les performances de traduction de GAMaT, nous comparons les traductions de GAMaT lancé avec les poids optimisés par MERT pour MOSES (*GAMaT-MERT*), avec les traductions de GAMaT lancé avec les poids obtenus des différentes exécutions de GAWO (*GAMaT+GAWO-i*), où i est le numéro du lancer.

Paire de langues	Système	BLEU ↑	TER ↓
TR-AN	**MOSES+MERT**	10,29	83,03
	GAMaT+MERT	6,46	82,05
	GAMaT+GAWO-1	8,81	80,21
	GAMaT+GAWO-2	8,57	80,39
	GAMaT+GAWO-3	**8.87**	**79,55**
	GAMaT+GAWO-4	8,72	80,32
	GAMaT+GAWO-5	8,67	80,88
	GAMaT-GAWO-moyen	$8{,}73_{\pm 0.1}$	$80{,}27_{\pm 0{,}41}$
FR-AN	**MOSES+MERT**	31,29	52,14
	GAMaT+MERT	24,84	57,18
	GAMaT+GAWO-1	28,78	53,83
	GAMaT+GAWO-2	**29,13**	**53,63**
	GAMaT+GAWO-3	28,79	53,68
	GAMaT+GAWO-4	28,83	53,70
	GAMaT+GAWO-5	29,08	53,60
	GAMaT-GAWO-moyen	$28{,}91_{\pm 0{,}14}$	$53{,}68_{\pm 0{,}07}$

TABLE 5.7 – Performances de traduction sur l'ensemble de test en matière de BLEU et TER.

Les performances de traduction obtenues montrent que l'optimisation des poids avec GAWO améliore considérablement les performances de traduction de GAMaT. En effet, selon le score BLEU, la métrique que nous utilisons au cours de l'optimisation, GAMaT fait mieux, en moyenne, qu'auparavant de plus de 2,2 points pour la paire TR-AN, et de plus de 4,0 points pour la paire FR-AN. Mais GAMaT ne dépasse pas les performances du système de référence MOSES. L'amélioration est également visible selon la métrique TER, où nous surpassons les résultats précédents de plus de 1,7 points et ceux de MOSES de 2,76 points pour la paire TR-AN. D'un autre côté, pour la paire FR-AN, nous dépassons les résultats précédents de plus de 3,5 points.

Comme nous l'avons expliqué précédemment, nous avons lancé GAWO cinq fois pour vérifier la stabilité du processus d'optimisation. À partir de ces cinq exécutions, nous avons lancé GAMaT avec l'ensemble des poids résultant de chaque exécution de GAWO. À partir des résultats obtenus (voir la tableau 5.7), nous avons calculé l'intervalle de confiance pour les scores BLEU et TER. Pour la paire de langues TR-AN, le rang du score BLEU est de 8,73 ± 0,1, et de 28,91 ± 0,14 pour la paire de langues FR-AN, avec une confiance de 95% pour les deux paires. Ces faibles intervalles démontrent que le comportement de GAWO est stable et qu'il permet à GAMaT d'obtenir les mêmes performances de traduction à chaque fois que nous l'exécutons.

L'important écart de performances de traduction entre les deux paires de langues, FR-AN et TR-AN, que nous pouvons constater à travers les résultats du Tableau 5.7, peut s'expliquer par deux aspects : le premier est la taille des données d'apprentissage, où pour la paire TR-AN nous n'avons que 200K phrases parallèles pour l'apprentissage contre 1,5M de phrases parallèles pour la paire FR-EN ; le deuxième est la complexité de la langue turque par rapport à la langue anglaise, où l'importante différence entre ces deux langues rend le processus de détection des alignements, entre mots et/ou segments, plus difficile. Afin d'analyser l'influence de ce deuxième aspect, nous créons un nouveau ensemble d'apprentissage pour la paire FR-AN avec le même nombre de phrases parallèles que pour la paire TR-AN. Cela nous permettra de voir à quel point

la complexité d'une langue peut influencer les performances de traductions.

Dans la tableau 5.8, nous présentons les performances de traduction des MOSES et celles de GAMaT, sur les mêmes ensembles de test présentés auparavant, avec des ensembles d'apprentissage de la même taille pour les deux paires de langues (FR-AN et TR-AN).

Paire de langues	Taille du corpus d'apprentissage	Système	BLEU ↑	TER ↓
TR-AN	**200K**	**MOSES-MERT**	10,29	83,03
		GAMaT-GAWO	28,91	80,41
FR-AN	**1.5M**	**MOSES-MERT**	31,29	52,14
		GAMaT-GAWO	28,91	53,68
	200k	**MOSES-MERT**	29,51	53,89
		GAMaT-GAWO	26,33	5517

TABLE 5.8 – Comparaison des performances de traduction entre les deux paires de langues (TR-AN, FR-AN) avec le même nombre de phrases parallèles pour l'apprentissage.

Comme nous pouvons le constater à travers ces résultats, en baissant la taille du corpus d'apprentissage des phrases parallèles, pour la paire FR-AN, de 1,5M phrases à 200K phrases, les performances de traduction baissent de 1,78 points en BLEU pour MOSES, et de 2,58 points pour GAMaT. Ces performances en TER, baissent de 1,75 points pour MOSES et de 1,45 pour GAMaT. Cette baisse de performances de traduction s'explique par le fait d'avoir réduit la taille des données d'apprentissage. Cependant, même avec le 200K phrases parallèles pour l'apprentissage, les performances de traduction pour la paire FR-EN restent nettement meilleures par rapport à celles de la paire TR-EN (voir Tableau 5.8. Nous pouvons en conclure que la qualité des données d'apprentissage pour la paire TR-EN, en particulier le fait que ces deux langues soient très différentes, est l'aspect le plus influent sur la dégradation des performances de traduction.

5.5 Performances de GAMaT optimisé par le réseau de neurones

La deuxième approche que nous avons proposée pour définir la fonction *fitness* de GAMaT est une approche basée sur un apprentissage neuronal, que nous avons présenté dans la Section 4.3 du Chapitre 4.2. L'idée de cette approche est d'apprendre une fonction pour prédire le score BLEU, ou le score TER, de l'hypothèse de traduction encodée dans un chromosome en fonction de ses huit scores. Dans cette section, nous présentons différentes expérimentations de l'utilisation de cette approche. En premier, nous détaillons, en quelques statistiques, les données générées pour apprentissage. Par la suite, nous présentons les performances d'apprentissage du réseau de neurones, où plusieurs configurations sont testées et évaluées. Au final, nous utilisons la fonction apprise comme fonction *fitness* dans GAMaT afin de traduire l'ensemble de test de la paire FR-AN et nous présentons la qualité des traductions en sortie.

5.5.1 Données d'apprentissage et de test

Afin d'évaluer cette approche, nous utilisons la paire de langues FR-AN qui nous permet de produire un grand nombre de données d'apprentissage, comparée à la paire TR-AN (voir la tableau 5.1). Pour la construction des données d'apprentissage, nous suivons le processus décrit dans la Section 4.3.2 du Chapitre 4, afin de produire un ensemble de paires $< v_i, \alpha_i >$, où v_i est l'ensemble des 8 scores d'évaluation du chromosome c_i et α_i le score BLEU de la traduction

encodée dans le chromosome c_i. Ces données sont séparées en deux corpus, un corpus d'apprentissage composé de 90% des données produites, et un corpus de test composé des 10% restantes.

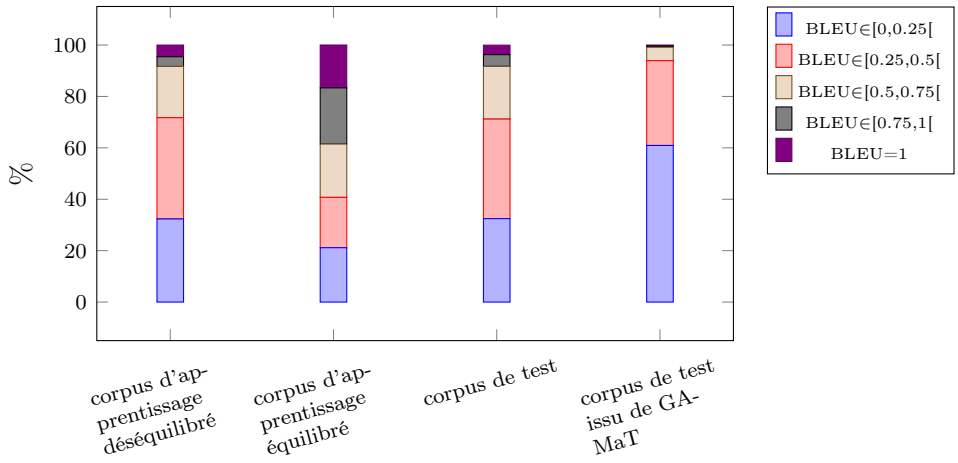


FIGURE 5.11 – Distribution des données dans les corpus d'apprentissage et de test, en matière de score BLEU.

Dans la figure 5.11, nous présentons la distribution des données du corpus d'apprentissage en fonction de scores BLEU des hypothèses encodées dans les chromosomes. Les chromosomes sont distribués sur cinq intervalles de score BLEU : [0,0.25[(très mauvaises traductions), [0.25,0.5[, [0.5,0,75[, [0,75,1[et les chromosomes avec score BLEU=1 (traductions parfaites). Dans la figure 5.11, nous présentons deux ensembles d'apprentissage et deux ensembles de test illustrés par différentes barres que nous listons comme suit :

— **corpus d'apprentissage déséquilibré :** ce corpus d'apprentissage est composé d'un grand nombre de chromosomes, générés par le processus présenté dans la Section 4.3 du Chapitre 4. La distribution de la qualité des traductions dans ce corpus est plus proche de la distribution réelle au cours du décodage. En effet, les traductions résultantes des fonctions d'initialisation de GAMaT et ses traductions en sortie sont en grande partie de faible et moyenne qualité par rapport aux traductions de référence.

— **corpus d'apprentissage équilibré :** ce corpus d'apprentissage est un sous-ensemble du précédent, où la distribution des données, en matière de score BLEU, est équilibrée entre les 5 classes du score BLEU. En effet, le fait d'avoir un grand nombre de traductions de mauvaise et moyenne qualité peut faire biaiser l'apprentissage du réseau de neurones et générer une fonction de prédiction du BLEU qui a tendance à ne prédire que des scores faibles. Ce corpus est généré en filtrant le précédent en imposant que chaque classe contienne le même nombre de chromosomes.

— **corpus de test :** ce corpus correspond au corpus d'apprentissage déséquilibré et il a la même distribution que lui. Évaluer le réseau de neurones sur un corpus de test pareil, peut être un inconvénient, car il n'existe que peu de bonnes traductions et le test se fera en grande partie sur de mauvaises traductions. Au moment du test, le réseau de neurones doit estimer la qualité des traductions avec la même précision, quelle que soit leur qualité. C'est pourquoi, nous avons également construit le corpus d'apprentissage équilibré (voir Figure 5.11).

— **corpus de test issu de GAMaT :** les distributions des données en matière de score BLEU, des corpus précédents, ne sont pas nécessairement respectées dans GAMaT, au cours du processus de décodage. Par conséquent, nous évaluons également la fonction apprise par le réseau de neurones sur un autre corpus de test représentatif des chromosomes au cours du décodage. Les chromosomes de ce corpus sont ceux de la population finale du processus de traduction de GAMaT. Le corpus contient 50K chromosomes construits à partir de 1 000 phrases sources, où pour chaque phrase source nous produisons 50 chromosomes. Il contient très peu de traductions parfaites et relativement plus de très mauvaises traductions par rapport au corpus de test précédent.

Dans ce qui suit, nous présentons différentes expérimentations en utilisant les deux corpus d'apprentissage et les deux corpus de test.

5.5.2 Métrique d'évaluation

Pour évaluer la précision de la fonction de prédiction apprise par le réseau de neurones, l'erreur absolue moyenne (EAM pour erreur absolue en moyenne) [Willmott and Matsuura, 2005] est utilisée pour estimer la différence (l'erreur) entre les prédictions et les scores BLEU réels sur les corpus de test, comme nous le présentons dans la formule 5.1.

$$MAE_{BLEU} = \frac{1}{n}\sum_{i=1}^{n} |BLEU_r(c_i) - BLEU_p(c_i)| \tag{5.1}$$

Où $BLEU_r(c_i)$ est le score BLEU réel de la traduction encodée dans le chromosome c_i et $BLEU_p(c_i)$ est le score BLEU prédit, pour la même traduction, par la fonction apprise par le réseau de neurones. La valeur n représente la taille de l'ensemble de test.

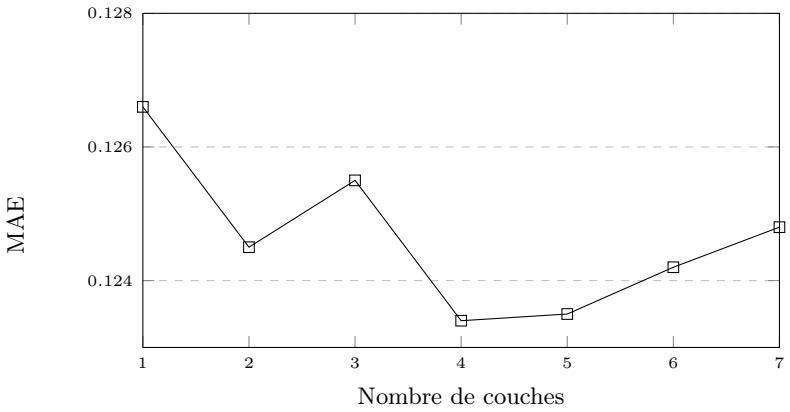


FIGURE 5.12 – L'influence du nombre de couches sur les performances du réseau de neurones.

5.5.3 Influence de l'architecture du réseau de neurones sur les performances

Dans cette section, nous étudions la performance du réseau de neurones en fonction du nombre de couches (1 à 7) et en fonction du nombre de neurones (8, 16, 32 et 64) dans chaque couche. Pour cela, nous faisons varier le nombre de couches de 1 à 7 et le nombre de neurones, par couche, entre 8, 16, 32 et 64. Dans la figure 5.12 et la figure 5.13, nous montrons les performances du

réseau de neurones en matière de score MAE, pour un apprentissage sur les corpus d'apprentissage et de test déséquilibrés, en fonction du nombre de couches et du nombre de neurones. Pour ces expérimentations nous utilisons 1.25M de données d'apprentissage.

Dans la courbe de la figure 5.12, chaque point est associé à un nombre de couches et sa valeur représente la performance moyenne (valeur MAE), sur l'ensemble de test, des performances avec les différentes valeurs du nombre de neurones. Dans la courbe de la figure 5.13, chaque point est associé à un nombre de neurones par couche et sa valeur représente la performance moyenne des performances avec les différentes valeurs du nombre de couches. À travers ces deux courbes, nous pouvons voir qu'en moyenne, sur les différentes configurations, les meilleures performances de la fonction de prédiction apprise par le réseau de neurones, sont obtenues avec 4 couches et 32 neurones par couche.

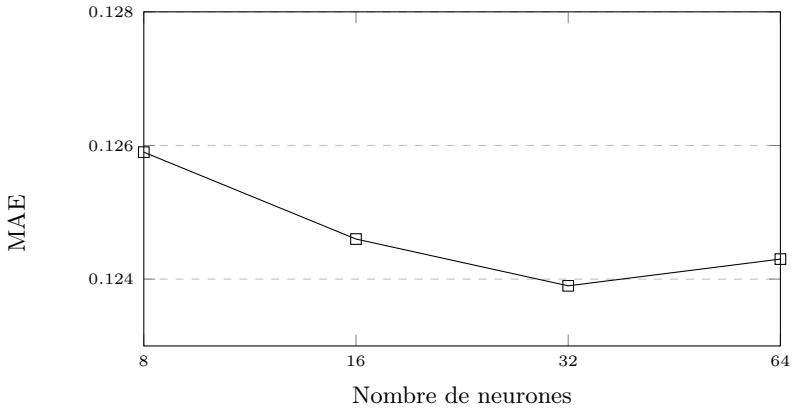


FIGURE 5.13 – L'influence du nombre de couches sur les performances du réseau de neurones.

Dans la figure 5.14, nous présentons les performances d'apprentissage, évaluées sur l'ensemble de test, de manière détaillée en fonction de ces deux paramètres. La meilleure performance est obtenue avec une configuration de 5 couches et 64 neurones par couche.

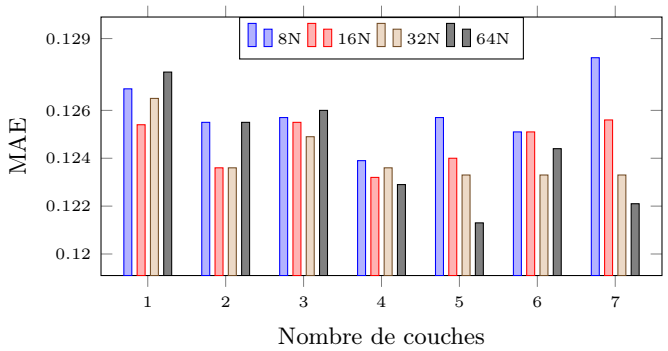


FIGURE 5.14 – Détails de l'influence du nombre de couches et du nombre de neurones sur les performances d'apprentissage su l'ensemble de test déséquilibré.

Sur la base de ces performances, nous décidons de prendre la configuration de 4 couches et 32 neurones pour la suite des expérimentions, car cette configuration est la meilleure en moyenne.

5.5.4 Influence de la taille et de la distribution du corpus d'apprentissage

Dans cette section, nous présentons les performances d'apprentissage en fonction de la taille du corpus d'apprentissage. Dans ce sens, nous augmentons le nombre de chromosomes de 225 000 à 2.5M dans les deux corpus d'apprentissage, déséquilibré (voir la figure 5.15) et équilibré (voir la figure 5.16).

Dans la figure 5.15, La partie gauche du graphique (a), montre les performances d'apprentissage en fonction de la taille du corpus d'apprentissage déséquilibré, sur les deux ensembles de test, déséquilibré et celui issu de GAMaT. Dans la partie droite du graphique (b), nous détaillons les performances d'apprentissage, en fonction de la taille du corpus, pour des sous-ensembles de test. En effet, nous avons créé, à partir de l'ensemble de test déséquilibré, 5 sous-ensembles en fonction des scores BLEU des traductions. Cela nous permet d'analyser le comportement de la fonction de prédiction en fonction de la qualité réelle de la traduction.

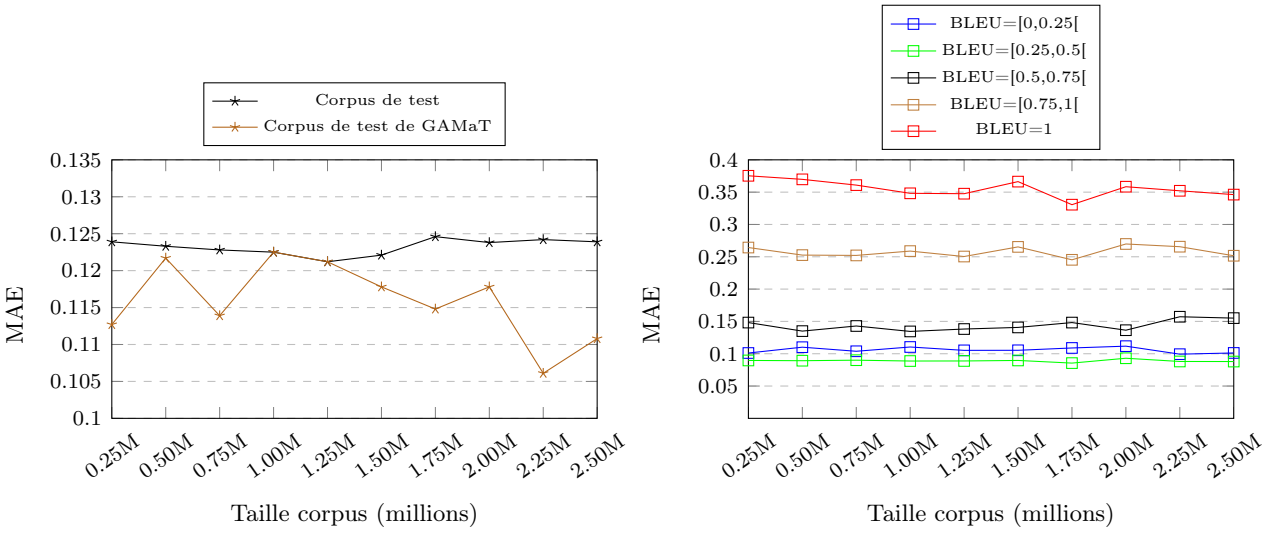
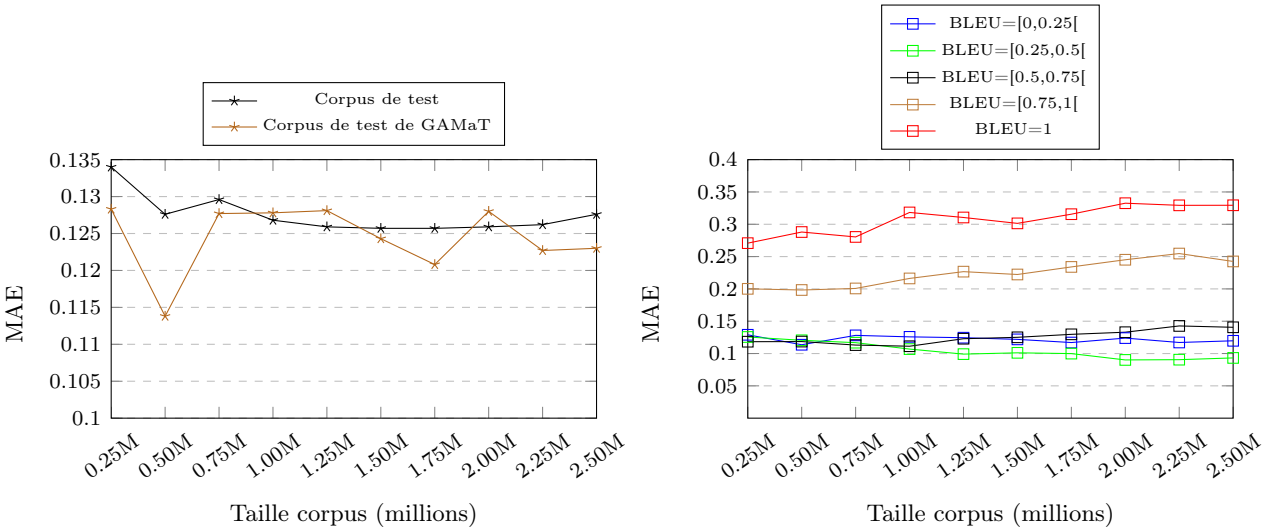


FIGURE 5.15 – L'influence du nombre de données pour le corpus d'apprentissage déséquilibré. Le graphique (a) représente l'évolution du MAE, sur les ensembles de test, en fonction de la taille du corpus d'apprentissage. Le graphique (b) représente la même évolution, mais sur les sous-ensembles de l'ensemble de test déséquilibré.

À partir de la figure 5.15, nous pouvons remarquer que la variation de taille du corpus d'apprentissage déséquilibré n'a pas d'influence visible sur la qualité de la fonction de prédiction sur l'ensemble de test déséquilibré, où l'erreur entre les scores BLEU réels et prédits varie faiblement entre 0,121 et 0,125. Sur l'ensemble de test issu de GAMaT, la courbe évolue de manière chaotique, mais nous pouvons remarquer que la qualité de la fonction d'estimation a tendance à être meilleure quand on augmente la taille du corpus, où la meilleure performance est obtenue avec un corpus d'apprentissage de 2,25M de données avec une erreur à 0,106.

Au niveau des sous-ensembles, la fonction d'estimation fait plus d'erreurs sur les traductions de bonne qualité, ayant un score BLEU supérieur à 0,75 et arrive à mieux prédire les traductions

ayant un faible score BLEU. Ce comportement est dû au fait que les mauvaises traductions sont beaucoup plus nombreuses dans le corpus d'apprentissage.



(a) (b)

FIGURE 5.16 – L'influence du nombre de données pour le corpus d'apprentissage équilibré. Le graphique (a) représente l'évolution du MAE, sur les ensembles de test, en fonction de la taille du corpus d'apprentissage. Le graphique (b) représente la même évolution, mais sur les sous-ensembles de l'ensemble de test déséquilibré.

Dans la figure 5.16, nous présentons les mêmes expériences, sauf que dans ce cas nous évaluons les performances du système appris en utilisant le corpus d'apprentissage équilibré. En équilibrant les données d'apprentissage, la fonction de prédiction perd en qualité. En effet, l'erreur de prédiction, estimée par le score MAE, dépasse en moyenne le 0,125, comparée à une erreur inférieure à 0,125 avec l'ensemble déséquilibré. Sur l'ensemble de test déséquilibré, la fonction de prédiction s'améliore en augmentant la taille du corpus d'apprentissage, tandis que la variation n'influence pas les performances sur l'ensemble de test issu de GAMaT, où l'évolution reste chaotique.

La dégradation de la qualité de la fonction de prédiction en utilisant un corpus d'apprentissage équilibré, s'explique par l'analyse du graphique (b) de la figure 5.16. En effet, comme les données sont équilibrées au moment de l'apprentissage, la fonction de prédiction améliore sa prédiction pour les traductions de bonne qualité (score BLEU supérieur à 0,75) mais cela n'est pas suffisant pour améliorer les performances globales car ces bonnes traductions ne sont pas nombreuses dans les ensembles de test comparées aux traductions de moins bonne qualité. Dans la figure 5.16-(b) nous remarquons que pour les traductions ayant un BLEU entre 0,75 et 1 le score EAM est entre 0,2 et 0,25, tandis qu'il est supérieur à 0,25 pour les mêmes traductions dans la figure 5.15-(b). Pour les données équilibrées, le score EAM est autour de 0,3 pour les traductions parfaites (BLEU=1), tandis qu'il est de 0,35 en moyenne pour les données d'apprentissage déséquilibrées. Pour les traductions ayant un score BLEU entre 0,25 et 0,5, les plus nombreuses dans le corpus de test (voir Figure 5.11), le score EAM est inférieure à 0,1 en moyenne pour les données d'apprentissage déséquilibrées (voir Figure 5.15-(b)), tandis qu'il est supérieur à 0,1 en

moyenne pour les données équilibrées (voir Figure 5.16-(b)).

5.5.5 Résultats finaux

Dans la tableau 5.9, nous résumons les performances de la fonction de prédiction apprise par le réseau de neurones. Dans ce tableau, nous présentons les performances, en score MAE, des configurations du réseau de neurones qui donnent les meilleurs résultats pour les deux corpus d'apprentissage et sur les deux corpus de test.

		Corpus de test	
Corpus d'apprentissage	**Taille du corpus**	**Test**	**GAMaT**
Déséquilibré	1,25M	0,12	0,12
	2,25M	0,12	0,10
Équilibré	1,50M	0,125	0,12
	0,50M	0,12	0,11

TABLE 5.9 – Meilleurs scores EAM pour les deux corpus d'apprentissage sur les deux corpus de test.

De manière générale, l'erreur de prédiction, sur les ensembles de test de 50K chromosomes, est très faible (0,12 en moyenne, en termes de MAE). La qualité de prédiction est meilleure avec un corpus d'apprentissage déséquilibré, où avec un corpus d'apprentissage de 2,25M de chromosomes, nous obtenons un score EAM de 0,1 sur l'ensemble de test issu de GAMaT. Pour l'ensemble de test déséquilibré, il ne faut que 1,25M de chromosomes pour avoir un score EAM de 0,12.

Même si les scores que nous venons de présenter sont faibles, la fonction de prédiction a des difficultés à prédire le score BLEU des hypothèses de traduction de bonne qualité, ayant un score supérieur à 0.75 (voir la figure 5.15-(b) et la figure 5.16-(b)). Cette faiblesse de prédiction peut fausser le processus de décodage si nous utilisons la fonction de prédiction du score BLEU comme fonction *fitness* pour GAMaT.

Afin d'étudier le comportement de cette fonction de prédiction au cours du processus de décodage de GAMaT, nous l'utilisons comme fonction *fitness* dans GAMaT pour évaluer les chromosomes. Nous traduisons l'ensemble de test de 1 000 phrases sources en français vers l'anglais. Cet ensemble de test est le même que celui utilisé pour évaluer les performances de traduction de GAMaT et de MOSES dans les sections précédentes. Dans la tableau 5.10, nous présentons les performances de traduction de GAMaT, en matière de score BLEU et score TER, en utilisant la nouvelle fonction *fitness* (GAMaT-RN), tout en les comparant aux performances précédentes de GAMaT.

Nous pouvons remarquer que l'utilisation de la fonction de prédiction du score BLEU ne permet pas d'améliorer les performances de traduction de GAMaT, qui sont même nettement moins bonnes que celles de GAMaT optimisé par GAWO, où le score BLEU est de 22,04 et le score TER de 63,42. Cette dégradation des performances de traduction peut s'expliquer par le fait que la fonction de prédiction n'arrive pas avoir la même précision de prédiction pour les bonnes hypothèses de traduction et les moyennes et mauvaises hypothèses. Par ailleurs, au cours

Paire de langues	Système	BLEU ↑	TER ↓
FR-AN	**MOSES-MERT**	31,29	52,14
	GAMaT-MERT	24,84	57,18
	GAMaT-GAWO	28,91	53,68
	GAMaT-RN	22,04	63,42

TABLE 5.10 – Comparaisons des performances de traduction de GAMaT en utilisant les deux fonctions *fitness*.

de l'apprentissage du réseau de neurones, nous utilisons le score EAM pour évaluer la qualité d'apprentissage, ce score étant la différence absolue moyenne, il ne prend pas en considération le principe d'ordre entre les chromosomes d'une population. Or, dans les mécanismes de sélection et de remplacement, les chromosomes doivent être correctement ordonnés afin de permettre à GAMaT de prendre les bonnes décisions au cours des mécanismes de sélection et de remplacement.

5.6 Résultats comparatifs

Dans cette dernière section, nous présentons différentes expérimentations afin d'étudier plus en détail les performances de traduction de GAMaT et aussi afin de comparer ses performances avec celles des performances de MOSES et celles d'un autre système de traduction neuronal. Dans la suite de cette section, nous prenons un ensemble de test de 5 000 phrases sources en français pour les traduire vers l'anglais avec les trois systèmes.

GAMaT, notre décodeur génétique est optimisé par GAWO, car c'est cet algorithme d'optimisation des poids qui aboutit aux meilleures performances pour GAMaT. Les poids de la fonction log-linéaire de MOSES, le décodeur de référence en traduction automatique statistique, sont optimisés par MERT. Pour le système de traduction neuronal, nous utilisons OpenNMT, un outil de mise en place de systèmes de traduction neuronaux. Le système NMT (pour Neural Machine Translation) est basé sur un encodeur-décodeur, comme présenté dans la Section 1.3.2 du Chapitre 1. Chaque unité est basée sur un réseau de neurones LSTM avec plusieurs couches cachées et utilisant différentes techniques d'attention [Menacer et al., 2017]. Pour l'apprentissage du système de traduction neuronal, nous utilisons les mêmes données de la paire FR-AN, présentée au début de ce chapitre.

Dans la tableau 5.11, nous présentons les performances de traduction des trois systèmes en matière de score BLEU et de score TER. Nous présentons également quelques détails sur les traductions en sortie des trois systèmes.

Système	BLEU	TER	taille moyenne des traductions	# moyen de segments
MOSES	28,45	56,62	22,7	9,0
NMT	**30,81**	**53,85**	21,8	-
GAMaT	26,31	58,39	22,7	8,5

TABLE 5.11 – Performances de traduction comparatives entre GAMaT, MOSES et NMT.

La première remarque est que la différence entre les performances de traduction de GAMaT

et de MOSES est du même ordre que la différence présentée auparavant, pour un ensemble de test de 1 000 phrases. Le système de traduction neuronal (NMT) propose les meilleures performances avec un score BLEU de 30,81, et un score TER de 53,85. En matière de taille de traduction en sortie des systèmes, MOSES et GAMaT, les deux systèmes à base d'apprentissage statistique, produisent des traductions avec la même taille moyenne qui est de 22,7 mots par traduction. Quant au système neuronal, il produit des traductions plus courtes, en moyenne 21.8 mots, cela peut être expliqué par le fait que ce système n'utilise pas de pénalité de la taille de la traduction et donc favorise les courtes hypothèses de traduction qui maximisent le score d'évaluation au cours du décodage. En matière de nombre de segments, MOSES produit des traductions avec un nombre de segments moyen de 9,1 segments par traduction, et 8,5 segments par traduction pour GAMaT. Les deux systèmes produisent quasiment le même nombre de segments en moyenne. NMT, le système neuronal, est basé sur une traduction à base de mots au cours du décodage, donc la notion de segmentation n'existe pas.

Dans la tableau 5.12, nous détaillons les performances de traduction afin d'analyser le comportement des trois systèmes en fonction de la taille des phrases sources. Nous créons quatre sous-ensembles de test, à partir de l'ensemble de 5 000 phrases sources, où chaque sous-ensemble est composé de 1 000 phrases. Le premier sous-ensemble est celui des phrases sources composées de moins de 15 mots, que nous considérons comme phrases courtes. Le deuxième est celui des phrases d'une taille entre 15 et 22 mots. Le troisième est celui des phrases d'une taille entre 22 et 32 mots. Le quatrième sous-ensemble est de celui des phrases longues, celles composées de plus de 32 mots. Nous avons fixé ces intervalles sur la base d'une étude statistique du corpus de test de base de 5 000 phrases, afin d'avoir 4 sous-ensembles contenant 1 000 phrases chacun.

Taille phrases	**GAMaT**		**MOSES**		**NMT**	
Métrique	**BLEU**	**TER**	**BLEU**	**TER**	**BLEU**	**TER**
Taille1 < 15	29,11	53,26	30,53	51,67	**32,52**	**49,43**
Taille2 < 22	27,65	56,59	30,24	54,41	**31,19**	**52,23**
Taille3 < 32	25,38	59,30	27,58	57,43	**30,49**	**54,36**
Taille4 > 32	26,10	59,36	28,24	57,75	**31,20**	**54,02**

TABLE 5.12 – Performances de traduction comparatives entre GAMaT, MOSES et NMT en fonction des tailles des phrases sources.

Pour les différentes tailles de phrases sources, le système de traduction neuronal a toujours les meilleures performances de traduction, en BLEU et en TER, comparé aux systèmes à base d'apprentissage statistique. Nous remarquons que pour les courtes phrases, les écarts entre les trois systèmes sont moins importants, sur la base des deux métriques d'évaluation. D'un autre côté, plus la taille augmente, plus NMT est performant que les deux autres systèmes, où sur le sous-ensemble des longues phrases, NMT dépasse MOSES par 3 points en BLEU et dépasse GAMaT par 5 points. En score TER, NMT fait mieux que MOSES et GAMaT par, respectivement, 3,7 points et 4,3 points. Cet écart est moins important sur les 5 000 phrases, où NMT dépasse MOSES et GAMaT par, respectivement, 2,2 points et 3,8 en BLEU, et les dépasse par 1,3 points et 4,5 points en TER.

En analysant manuellement, phrase par phrase, les traductions en sortie des trois systèmes, nous observons que les différences entre les traductions de ces systèmes sont moins importantes

que ce que les scores des métriques montrent au niveau du corpus. Sur la base de ces observations, nous décidons d'analyser les performances des trois systèmes plus en détail en évaluant les traductions phrase par phrase. À cette fin, nous utilisons à nouveau les métriques BLEU et TER pour évaluer les traductions des trois systèmes phrase par phrase. Ayant un score pour chaque traduction, nous comparons de manière plus approfondie les performances des trois systèmes. Mieux encore, nous pouvons combiner, a posteriori, les traductions de ces systèmes et étudier à quel point les performances globales sur l'ensemble de test de 5 000 phrases peuvent être améliorées.

Afin de mettre en palace cette combinaison oracle, nous prenons l'ensemble des 5 000 phrases sources $f = \{f_1, ..., f_{5000}\}$ et les ensembles de traductions des trois systèmes ($e^{\text{NMT}} = \{e_1^{\text{NMT}}, ..., e_{5K}^{\text{NMT}}\}$, $e^{\text{GAMaT}} = \{e_1^{\text{GAMaT}}, ..., e_{5K}^{\text{GAMaT}}\}$, $e^{\text{MOSES}} = \{e_1^{\text{MOSES}}, ..., e_{5K}^{\text{MOSES}}\}$). Pour chaque phrase source f_i, nous sélectionnons la traduction ayant le meilleur score BLEU/TER parmi les trois traductions (e_i^{NMT}, e_i^{GAMaT}, e_i^{MOSES}) des trois systèmes combinés. De cette manière, nous construisons un nouvel ensemble de traductions, composé de 5 000 traductions, que nous évaluons au niveau du corpus par la métrique utilisée dans cette combinaison oracle. Dans la tableau 5.13 et la tableau 5.14, nous montrons tous les résultats détaillés des combinaisons en utilisant, respectivement, les scores BLEU et TER. Nous commençons par combiner les trois systèmes deux-à-deux, puis les trois systèmes à la fois. Pour chaque combinaison, nous calculons le pourcentage (%) des traductions sélectionnées à partir de chaque système dans la construction du nouvel ensemble de traduction.

Système	**BLEU**	**# MOSES**	**# NMT**	**# GAMaT**	**# égal**
MOSES	28,45	5000	-	-	-
NMT	30,81	-	5000	-	-
GAMAT	26,31	-	-	5000	-
MOSES-GAMaT	29,91	$2481_{(49,62\%)}$	-	$1556_{(31,12\%)}$	$963_{(19,26\%)}$
NMT-GAMaT	33,06	-	$2978_{(59,56\%)}$	$1651_{(33,02\%)}$	$371_{(07,42\%)}$
MOSES-NMT	33,77	$1876_{(37,52\%)}$	$2620_{(52,40\%)}$	-	$504_{(10,08\%)}$
MOSES-NMT-GAMaT	34,32	$1630_{(32,60\%)}$	$2333_{(46,66\%)}$	$768_{(15,36\%)}$	$269_{(05,38\%)}$

TABLE 5.13 – Comparaison et combinaison oracle des trois systèmes au niveau de la phrase en utilisant la métrique BLEU.

De manière globale, nous observons que toutes les combinaisons du Tableau 5.13, permettent une importante amélioration des performances de traduction des systèmes combinés. En effet, lorsque nous combinons notre système GAMaT avec MOSES, nous surpassons les performances du meilleur de ces deux systèmes (MOSES) de 1,45 points en BLEU. La combinaison de GAMaT avec le système neuronal NMT surpasse les performances de ce dernier de 2,25 points. Grâce à ces deux cas de combinaison, en plus d'améliorer le score BLEU global sur l'ensemble de test, GAMaT parvient à faire mieux que MOSES dans 31.12% des cas. Il obtient les mêmes performances que MOSES dans 19.26% des cas. Comparé à NMT, il obtient de meilleures performances dans 33,02% des cas, et fait de même dans 7,42% des cas. La combinaison de MOSES avec NMT permet d'obtenir une performance globale de 33,77 en BLEU, ce qui est légèrement supérieur à la combinaison entre NMT et GAMaT (33,06 de BLEU). Dans cette combinaison, le système neuronal fait mieux dans 52,4% des traductions contre à 37,52% pour MOSES.

La dernière combinaison, *"MOSES-NMT-GAMaT"* dans la Tableau 5.13, consiste à combiner les trois systèmes en même temps. Nous observons que cette combinaison permet d'obtenir les meilleures performances de traduction sur l'ensemble de test. En effet, cette combinaison donne

un score BLEU de 34,32, qui est supérieur au système NMT de plus de 3,5 points. Dans cette combinaison globale, notre système génétique GAMaT fait mieux que les deux autres systèmes dans 15,36% des cas et il obtient les mêmes performances dans 5,38% des traductions.

Les résultats du Tableau 5.14 présentent les mêmes expériences de combinaisons que dans la tableau précédent, sauf que, dans ce cas, nous utilisons les scores TER pour évaluer les traductions au niveau de la phrase. La métrique TER est connue pour être meilleure dans l'évaluation des traductions au niveau de la phrase par rapport à la métrique BLEU, qui est généralement utilisée au niveau du corpus.

Système	**BLEU**	**# MOSES**	**# NMT**	**# GAMaT**	**# égal**
MOSES	56,62	5000	-	-	-
NMT	53,85	-	5000	-	-
GAMaT	58,39	-	-	5000	-
MOSES-GAMaT	54,28	$2018_{(40,36\%)}$	-	$1222_{(24,44\%)}$	$1760_{(35,2\%)}$
NMT-GAMaT	50,69	-	$2707_{(54,14\%)}$	$1263_{(25,26\%)}$	$1030_{(20,6\%)}$
MOSES-NMT	50,21	$1480_{(29,60\%)}$	$2367_{(47,34\%)}$	-	$1153_{(23,06\%)}$
MOSES-NMT-GAMaT	49,43	$1677_{(33,54\%)}$	$2254_{(45,08\%)}$	$517_{(10,34\%)}$	$552_{(11,04\%)}$

TABLE 5.14 – Comparaison et combinaison oracle des trois systèmes au niveau de la phrase en utilisant la métrique TER.

Comme pour le BLEU, toutes les combinaisons avec les scores TER permettent de surpasser les performances de traduction des trois systèmes de base. Dans les combinaisons deux-à-deux, les performances de la combinaison de GAMaT avec les deux autres systèmes est supérieure aux performances de traduction de MOSES et celles de NMT, respectivement, de 2,34 points et 3,16 points, en TER. Dans ces combinaisons, GAMaT obtient de meilleures performances que MOSES dans 24,44% des cas et atteint les mêmes performances dans 35,2% des cas. GAMaT fait mieux que NMT dans 25,26% et obtient les mêmes performances dans 20,6% des traductions. La combinaison des trois systèmes permet d'obtenir une performance de traduction de 49,43% en TER, qui surpasse le meilleur système (NMT) de plus de 4 points de TER. Dans ce cas, notre système génétique GAMaT obtient de meilleures performances que les deux autres dans 10,34% des cas et réalise les mêmes performances dans 11,04% des cas.

Ces expérimentations de combinaison détaillées montrent qu'il y a une complémentarité entre les trois différents systèmes de traduction. À travers ces résultats, nous pouvons conclure qu'il n'existe pas de système parfait, parmi les trois, qui réalise toujours de meilleures performances de traduction que les autres, même si le système neuronal est meilleur de manière globale sur un ensemble de test.

5.7 Conclusion et discussion

Dans ce chapitre, nous avons présenté un ensemble d'expérimentations afin d'évaluer le comportement et les performances de notre système de traduction à base d'algorithmes génétiques. Dans un premier temps, nous avons étudié le déroulement génétique du décodeur GAMaT, qui s'est avéré correct et qui permet de faire évoluer la population des chromosomes vers une population meilleure que celle à l'initialisation. Les paramètres du processus génétique de GAMaT

ont été fixés et optimisés afin d'avoir les meilleures performances possibles. Dans un second temps, nous avons présenté différents résultats de l'optimisation de la fonction *fitness* de GAMaT. Comme il a été présenté dans le Chapitre 4, nous avons proposé deux approches pour optimiser cette fonction qui combine 8 scores d'évaluation différents. Donc, nous avons évalué les performances de traduction de GAMaT avec une fonction *fitness* basée sur l'approche log-linéaire dont les poids sont optimisés par un autre algorithme génétique (GAWO). Nous avons aussi présenté des expérimentations de la mise en place de la deuxième solution pour l'optimisation de la *fitness* de GAMaT, qui est l'utilisation d'un réseau de neurones pour l'apprentissage d'une fonction qui prédit le score BLEU des traductions encodées dans les chromosomes.

Tout au long de ce chapitre, en plus des expérimentations réalisées pour l'analyse du comportement de GAMaT, associé aux deux fonctions *fitness*, nous avons présenté des résultats comparatifs afin de positionner notre système génétique par rapport aux systèmes de référence en traduction automatique. Une analyse comparative détaillée a été présentée en fin de chapitre entre notre système GAMaT, MOSES le système de référence en traduction statistique, et un autre système de traduction neuronal.

L'application de GAWO pour l'optimisation des poids de la fonction log-linéaire de GAMaT, a montré son efficacité. En effet, les performances de traduction de GAMaT se sont nettement améliorées, pour les deux paires de langues, français-anglais (FR-AN) et turc-anglais (TR-AN). Cette optimisation a permis de mieux adapter la fonction *fitness* au processus génétique de GAMaT, ce qui améliore ses prises de décision au cours des mécanismes de sélection et de remplacement, les deux mécanismes qui assurent la bonne convergence d'un processus génétique. Malgré cela, les performances de GAMaT restent inférieures à celles de MOSES en matière des scores BLEU et TER.

L'utilisation d'un réseau de neurones pour l'apprentissage d'une nouvelle fonction *fitness* qui prédit le score BLEU a montré des limites quand nous l'utilisons dans le processus de traduction de GAMaT. Les expérimentations et performances de l'apprentissage neuronal, que nous avons présentés sont très intéressants. En effet, la fonction de prédiction arrive à faire un score moyen d'erreur absolue de 0,11, sur un ensemble de test de 50K chromosomes. Cependant, l'application de cette fonction au cours du décodage de GAMaT n'a pas pu améliorer le processus d'évaluation, ce qui a détérioré la qualité des prises de décision. Cette nouvelle fonction de *fitness*, a montré des limites dans la prédiction du bon score BLEU des hypothèses de bonne qualité et aussi dans la capacité à bien ordonner les chromosomes d'une population.

L'analyse détaillée des traductions des trois systèmes, GAMaT, MOSES et NMT, au niveau de la phrase, montre que notre système de traduction génétique, GAMaT optimisé par GAWO, arrive à surpasser les performances de traduction des deux autres systèmes dans un certain nombre de traductions. Cette supériorité de GAMaT ne pouvait être détectée avec une évaluation classique au niveau du corpus sur l'ensemble de test. Par ailleurs, l'importante amélioration des performances de traduction après l'application des combinaisons Oracle, nous permet d'envisager de mettre en place un système combinant les trois systèmes de traduction de manière automatique sur la base de la structure de la phrase source.

6

Conclusion et perspectives

6.1 Conclusion

Dans ce manuscrit de thèse, nous avons présenté nos travaux de recherche consacrés à l'utilisation des algorithmes bio-inspirés dans le cadre de la traduction automatique. Notre principale contribution est l'application des algorithmes génétiques à la traduction automatique statistique. Nous proposons un système de TAS à base de segments avec un décodeur entièrement basé sur des algorithmes génétiques.

Dans ce cadre, nous avons présenté GAMaT *"Genetic Algorithm for Machine Translation"*, un algorithme génétique pour résoudre le problème de décodage pour la construction des traductions. Dans cet algorithme, la difficulté était l'adaptation et la redéfinition de toutes les composantes d'un algorithme génétique pour le problème de décodage d'un système de TAS à base de segments.

Dans le Chapitre 3, nous avons expliqué en détail nos choix de conception de GAMaT, en commençant par la représentation d'un chromosome qui encode un ensemble d'éléments d'une solution (segmentations de la source et de la cible, les alignements entre segments et l'hypothèse de traduction). Nous avons défini quatre fonctions d'initialisation, pour la production d'une population initiale de chromosomes, qui diffèrent selon la stratégie de segmentation et celle de la génération des segments cibles. Les fonctions de reproduction, à savoir le croisement et la mutation ont été complètement redéfinies pour les adapter à ce problème. Nous avons présenté une fonction de croisement qui arrive à faire croiser deux chromosomes en échangeant une même portion des deux chromosomes, et cela malgré le fait que les chromosomes dans GAMaT n'ont pas la même taille ce qui rend les fonctions de croisement classiques inapplicables. Pour la mutation, nous avons défini quatre fonctions modifiant les chromosomes au niveau des paires de segments : mutation par fusion de segments, mutation par séparation d'un segment, mutation par remplacement de la traduction d'un segment et la mutation par échange de position de deux segments cibles.

La moitié de nos contributions dans ce travail de thèse, est consacrée à la fonction d'évaluation des chromosomes de GAMaT, ou fonction *fitness*, qui permet d'estimer la qualité des solutions au cours du décodage génétique et afin de le guider vers de bonnes solutions finales. Nous avons proposé deux approches pour définir cette fonction *fitness*, que nous avons présentées dans le Chapitre 4.

La première proposition, est l'utilisation de la fonction log-linéaire pondérée, largement utilisée dans la littérature pour l'évaluation des hypothèses de traduction en combinant différents scores estimant plusieurs aspects d'une solution. Nous proposons un autre algorithme génétique, GAWO *"Genetic Algorithm for Weights Optimization"*, pour l'optimisation des poids de la fonction log-linéaire. Dans GAWO, nous proposons une implémentation classique des algorithmes génétiques manipulant des vecteurs de valeurs numériques. L'application de GAWO pour l'optimisation de la fonction *fitness* de GAMaT, nous a permis d'avoir un décodeur dans lequel, tous les problèmes d'optimisation sont traités à l'aide d'algorithmes génétiques.

Pour la deuxième proposition, nous sommes allés plus loin en remplaçant la fonction log-linéaire par une nouvelle fonction de prédiction du score BLEU. Cette fonction est apprise à l'aide d'un réseau de neurones qui met en corrélation les différents scores d'un chromosome (solution), combinés auparavant par la fonction log-linéaire, et le score BLEU de l'hypothèse de traduction encodée dans le chromosome en question. D'un côté, cette fonction a pour objectif de trouver un meilleur moyen de combiner les différents scores que celui la combinaison *via* la fonction log-linéaire, qui nécessite une optimisation des poids associés aux scores. D'un autre côté, avec une fonction *fitness* qui prédit le score d'une métrique d'évaluation, le BLEU dans ce cas, d'une hypothèse de traduction au cours du décodage, nous espérions guider le processus de décodage génétique vers des traductions qui se rapprochent des traductions de référence afin d'améliorer les performances de traduction.

À la fin de ce manuscrit (Chapitre 5), nous avons présenté un ensemble d'expérimentations que nous avons réalisées pour analyser le comportement de GAMaT, le décodeur génétique, et également le comportement de GAWO, l'autre algorithme génétique pour l'optimisation des poids appliqué à GAMaT. Nous avons présenté aussi, différentes expérimentations pour analyser les performances d'apprentissage du réseau de neurones pour la définition de la fonction de prédiction du score BLEU. Par la suite, nous avons présenté les performances de traduction de notre système génétique et cela en utilisant, comme *fitness*, la fonction log-linéaire optimisée par GAWO et la fonction de prédiction du score BLEU. Nous avons appliqué cette évaluation des performances sur des ensembles de test dans deux paires de langues différentes, à savoir la paire français-anglais (FR-AN) et la paire turc-anglais (TR-AN). En fin de chapitre, nous avons présenté des résultats comparatifs détaillés, afin de situer notre système par rapport à deux systèmes de référence en traduction automatique, à savoir MOSES et un système de traduction neuronal (NMT).

Les résultats de ces expérimentations nous ont montré que le comportement de GAMaT avec la fonction log-linéaire comme *fitness* est correct au sens où le processus de décodage génétique réussit à améliorer la qualité la population initiale de chromosomes et à converger vers une population optimale, et parvient à produire au final une traduction de meilleure qualité que celles qui existaient au début du processus.

L'application de GAWO sur GAMaT, pour l'optimisation des poids de la fonction log-linéaire, permet de définir une fonction *fitness* mieux adaptée au processus de décodage génétique de GAMaT, ce qui lui permet de produire de meilleures traductions en sortie. Sur les ensembles de test, pour les deux paires de langues, les performances de traduction de GAMaT se sont nettement améliorées en appliquant GAWO comparées aux performances de GAMaT avec des poids issus d'une optimisation faite sur MOSES. En effet, pour la paire FR-AN, nous avons noté une amé-

lioration de 4,0 points en BLEU et une amélioration de 3,5 en TER. Pour la paire TR-AN, nous avons noté une amélioration de 2,27 en BLEU et 1,78 en TER.

Nous avons testé plusieurs configurations du réseau de neurones, en variant le nombre de couches et le nombre de neurones, pour évaluer les performances d'apprentissage de la fonction de prédiction du score BLEU. En plus de la configuration, nous avons varié les données d'apprentissage pour évaluer ces performances en fonction de la qualité du corpus d'apprentissage et de sa taille. Les meilleures performances d'apprentissage que nous avons obtenues sont de 0,11 en EAM, un score qui estime la différence entre les vrais scores BLEU des hypothèses de traduction et les scores prédits par la fonction, sur un ensemble de test. Malgré cette faible différence, l'utilisation de cette fonction comme *fitness* dans GAMaT, n'arrive pas améliorer les performances de traduction, au contraire elles se sont dégradées comparées aux performances de GAMaT avec la fonction *fitness* optimisée par GAWO.

La comparaison des performances de traduction de notre système génétique avec celles des deux systèmes de l'état de l'art, a montré que, de manière globale, sur l'ensemble de test, notre système propose des traductions ayant des performances inférieures à celles de MOSES et NMT, ce dernier, basé sur une approche neuronale, proposant les meilleures performances. De manière détaillée, en comparant les traductions phrase par phrase, selon leur score BLEU, nous avons constaté que notre système génétique arrive à faire mieux que MOSES dans 31% des cas et mieux que NMT dans 32% des cas. Cette analyse détaillée, nous a permis de remarquer qu'il n'y a pas de système meilleur que les autres dans la traduction de toutes les phrases sources, même si les performances globales, sur le corpus, favorise un des systèmes par rapport aux autres.

6.2 Perspectives futures

Dans ce travail de thèse, nous avons présenté trois propositions, à savoir : un algorithme génétique pour le décodage, un autre algorithme génétique pour l'optimisation des poids de la fonction log-linéaire et au final, une fonction de prédiction du score BLEU comme *fitness* pour GAMaT. Nous avons vu que l'utilisation des algorithmes génétiques pour la traduction automatique statistique au niveau du décodeur, permet de mettre en place un système de traduction qui arrive à rivaliser avec les systèmes de l'état de l'art dans plus de 30% des traductions sur un ensemble de 5 000 phrases à traduire. Nous avons constaté aussi, que le fait de manipuler des hypothèses de traduction complètes au cours du décodage permet de proposer de nouvelles approches pour l'évaluation des solutions dans le but d'améliorer la qualité du décodage. Nos propositions et les résultats que nous avons présentés, peuvent ouvrir la porte à d'autres pistes de recherche.

Dans ce qui suit, nous listons les perspectives, de ce travail, que nous estimons intéressantes et prometteuses afin d'améliorer les performances de traduction et aussi afin d'élargir l'utilisation des algorithmes génétiques pour la traduction automatique.

La première perspective, que nous estimons très importante, est de revoir la conception de l'algorithme génétique de décodage (GAMaT), afin de trouver une autre représentation des hypothèses de traduction permettant l'application des fonctions de reproduction classiques dans la littérature des algorithmes génétiques. En effet, dans un processus de recherche génétique, le comportement global de l'algorithme et celui de ses fonctions doivent rester aléatoires : c'est à la

fonction *fitness* de guider le processus de décodage vers les solutions optimales. Or, dans ce que nous avons proposé, la structure du chromosome est complexe, contenant plusieurs éléments, ce qui nous oblige à contraindre les fonctions d'initialisation et de reproduction.

Une solution au problème précédent, est l'application de l'algorithme génétique au niveau du décodeur d'un système de traduction neuronal, et non pas un système de TAS à base de segments. En effet, le fait que nous utilisons dans GAMaT, une table de traduction à base de segments, pour la construction des traductions, nous a obligé à fixer cet encodage des solutions et à contraindre les fonctions génétiques afin de ne pas produire des solutions en dehors de l'espace de recherche. Dans un système de traduction neuronal, le décodage est à base de mots, où les hypothèses de traduction sont construites de manière incrémentale, comme dans MOSES, mais en rajoutant à chaque étape de la construction un mot et non pas un segment. Concevoir un algorithme génétique de décodage à base de mots, facilitera la représentation des solutions en chromosomes et facilitera par conséquent la définition des fonctions d'initialisation et de reproduction afin de mettre en place un processus génétique moins contraint. Cette proposition nous obligera à définir une nouvelle fonction *fitness* adaptée au problème de décodage d'un système de traduction neuronal.

De fait, dans ce que nous avons présenté, nous avons vu que la définition de la bonne fonction *fitness* de GAMaT est une tâche essentielle pour la construction d'un décodeur efficace. La manipulation d'hypothèses de traduction complètes tout au long du processus de décodage, nous permet de proposer de nouveaux scores à combiner dans la fonction log-linéaire ou même de la remplacer comme nous l'avons fait avec la fonction de prédiction du BLEU.

Comme possible travail futur sur la *fitness*, nous pensons à combiner notre système avec d'autres approches de traduction. L'idée est de rajouter de nouveaux scores d'évaluation des chromosomes de GAMaT, dans la fonction log-linéaire, ces scores étant issus d'évaluation faites par des systèmes de différentes approches. En effet, nous pouvons rajouter un score calculé par la fonction d'évaluation des hypothèses de traduction d'un système de traduction neuronal, qui prend en entrée la phrase source et l'hypothèse encodée dans le chromosome. L'autre possibilité est d'utiliser un système de traduction à base de règles de transfert pour analyser l'hypothèse de traduction. L'idée est de comparer la structure syntaxique de l'hypothèse par rapport à la structure syntaxique de la source afin d'estimer un nouveau score estimant la forme syntaxique de l'hypothèse, à rajouter dans la combinaison.

La fonction de prédiction du score BLEU des hypothèses de traduction, que nous avons utilisée comme *fitness* dans GAMaT, entre dans le cadre des travaux de l'estimation de la qualité des traductions [Langlois, 2015]. Dans ce domaine, l'idée est de mettre en place un mécanisme d'évaluation de la qualité des traductions en sortie d'un système sans les comparer aux traductions de référence. Au cours du décodage de GAMaT, nous manipulons des traductions complètes et les traductions de référence ne sont pas disponibles, donc nous pensons à adapter ou mettre en place un mécanisme d'estimation de la qualité des traductions afin de l'utiliser pour définir une nouvelle fonction *fitness* évaluant les chromosomes.

Bibliographie

[Al-Onaizan et al., 1999] Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, D., Och, F. J., Purdy, D., Smith, N. A., and Yarowsky, D. (1999). Statistical Machine Translation. In *Final Report, JHU Summer Workshop*, volume 30.

[Araujo, 2007] Araujo, L. (2007). How Evolutionary Algorithms are Applied to Statistical Natural Language Processing. *Artificial Intelligence Review*, 28(4) :275–303.

[Arnold et al., 1993] Arnold, D. J., Balkan, L., Meijer, S., Humphreys, R. L., and Sadler, L. (1993). *Machine Translation : an Introductory Guide*. Blackwells-NCC, London.

[Back, 1996] Back, T. (1996). *Evolutionary Algorithms in Theory and Practice : Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv :1409.0473*.

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (2005). METEOR : an Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

[Beyer and Schwefel, 2002] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution Strategies-A Comprehensive Introduction. *Natural Computing*, 1(1) :3–52.

[Binitha and Sathya, 2012] Binitha, S. and Sathya, S. S. (2012). A Survey of Bio Inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSCE)*, 2.

[Boitet, 1988] Boitet, C. (1988). Pros and Cons of the Pivot and Transfer Approaches in Multilingual Machine Translation. *Readings in Machine Translation*, pages 273–279.

[Bojar et al., 2017] Bojar, O., Buck, C., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Kreutzer, J., Logacheva, V., Monz, C., Negri, M., Névéol, A., Neves, M., Post, M., Riezler, S., Sokolov, A., Specia, L., Turchi, M., and Verspoor, K., editors (2017). *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics, Copenhagen, Denmark.

[Bojar et al., 2015] Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

[Bonabeau et al., 1999] Bonabeau, E., Marco, Dorigo, M., Théraulaz, G., Theraulaz, G., et al. (1999). *Swarm Intelligence : from Natural to Artificial Systems*. Number 1. Oxford University Press.

[Bossard and Rodrigues, 2015] Bossard, A. and Rodrigues, C. (2015). Une Approche Évolutionnaire pour le Résumé Automatique. In *TALN 2015-22ème Conférence sur le Traitement Automatique des Langues Naturelles*, volume 22.

[Brown et al., 1990] Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A Statistical Approach to Machine Translation. *Computational linguistics*, 16(2) :79–85.

[Brown et al., 1991] Brown, P. F., Lai, J. C., and Mercer, R. L. (1991). Aligning Sentences in Parallel Corpora. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, ACL '91, pages 169–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Brown et al., 1993] Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation : Parameter Estimation. *Comput. Linguist.*, 19(2) :263–311.

[Callison-Burch et al., 2010] Callison-Burch, C., Koehn, P., Monz, C., Peterson, K., Przybocki, M., and Zaidan, O. F. (2010). Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53. Association for Computational Linguistics.

[Chen and Zhu, 2008] Chen, H. and Zhu, Y. (2008). Optimization Based on Symbiotic Multispecies Coevolution. *Applied Mathematics and Computation*, 205(1) :47–60.

[Chen and Goodman, 1999] Chen, S. F. and Goodman, J. (1999). An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech & Language*, 13(4) :359–394.

[Chiang et al., 2011] Chiang, D., DeNeefe, S., and Pust, M. (2011). Two Easy Improvements to Lexical Weighting. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies : short papers-Volume 2*, pages 455–460. Association for Computational Linguistics.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv :1412.3555*.

[Črepinšek et al., 2013] Črepinšek, M., Liu, S.-H., and Mernik, M. (2013). Exploration and Exploitation in Evolutionary Algorithms : a Survey. *ACM Computing Surveys (CSUR)*, 45(3) :35.

[Devlin et al., 2014] Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1370–1380.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant System : Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1) :29–41.

[Douib et al., 2016] Douib, A., Langlois, D., and Smaïli, K. (2016). Genetic-based Decoder for Statistical Machine Translation.

[Douib et al., 2017a] Douib, A., Langlois, D., and Smaïli, K. (2017a). GAWO : Genetic-based Optimization Algorithm for SMT. In *ICNLSSP 2017-International Conference on Natural Language, Signal and Speech Processing*.

[Douib et al., 2017b] Douib, A., Langlois, D., and Smaïli, K. (2017b). A Translation Evaluation Function Based on Neural Network. *Schedae Informaticae*, 2016(Volume 25) :139–151.

[Echizen-ya et al., 1996] Echizen-ya, H., Araki, K., Momouchi, Y., and Tochinai, K. (1996). Machine Translation Method Using Inductive Learning with Genetic Algorithms. In *Proceedings of the 16th Conference on Computational Linguistics*, volume 2, pages 1020–1023. Association for Computational Linguistics.

[Elman, 1990] Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, 14(2) :179–211.

[Fogel, 2006] Fogel, D. B. (2006). Foundations of Evolutionary Computation. In *Modeling and Simulation for Military Applications*, volume 6228, page 622801. International Society for Optics and Photonics.

[Glover, 1989] Glover, F. (1989). Tabu Search—Part I. *ORSA Journal on Computing*, 1(3) :190–206.

[González, 2012] González, J. (2012). A Finite-state Approach to Phrase-based Statistical Machine Translation. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 90–98.

[Han and Wong, 2016] Han, A. L.-F. and Wong, D. F. (2016). Machine Translation Evaluation : a Survey. *arXiv preprint arXiv :1605.04515*.

[Han, 2001] Han, B. (2001). Building a Bilingual Dictionary with Scarce Resources : a Genetic Algorithm Approach. In *Proceedings of the Student Research Workshop, the Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

[Hasler et al., 2011] Hasler, E., Haddow, B., and Koehn, P. (2011). Margin Infused Relaxed Algorithm for MOSES. *The Prague Bulletin of Mathematical Linguistics*, 96 :69–78.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-term Memory. *Neural Computation*, 9(8) :1735–1780.

[Holland, 1973] Holland, J. H. (1973). Genetic Algorithms and the Optimal Allocation of Trials. *SIAM Journal on Computing*, 2(2) :88–105.

[Hopkins and May, 2011] Hopkins, M. and May, J. (2011). Tuning as Ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics.

[Hutchins, 1995] Hutchins, W. J. (1995). Machine Translation : a Brief History. In *Concise History of the Language Sciences*, pages 431–445. Elsevier.

[Hutchins, 2001] Hutchins, W. J. (2001). Machine Translation over Fifty Years. *Histoire, Epistemologie, Langage, Tome XXII, FASC*, 23 :7–31.

[Hüe, 1997] Hüe, X. (1997). Genetic Algorithms for Optimisation : Background and Applications. *The University of Edinburgh*.

[Katz, 1987] Katz, S. (1987). Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3) :400–401.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948.

[Klein et al., 2017] Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). Opennmt : Open-source Toolkit for Neural Machine Translation. *arXiv preprint arXiv :1701.02810*.

[Kocur and Bojar, 2016] Kocur, V. and Bojar, O. (2016). Particle Swarm Optimization Submission for WMT16 Tuning Task. In *Proceedings of the First Conference on Machine Translation*, pages 518–524, Berlin, Germany. Association for Computational Linguistics.

[Koehn, 2004] Koehn, P. (2004). Pharaoh : a Beam Search Decoder for Phrase-based Statistical Machine Translation Models. *Machine translation : From Real Users to Research*, pages 115–124.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). MOSES : Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

[Koehn and Knowles, 2017] Koehn, P. and Knowles, R. (2017). Six Challenges for Neural Machine Translation. *arXiv preprint arXiv :1706.03872*.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

[Koza, 1994] Koza, J. R. (1994). Genetic Programming as a Means for Programming Computers by Natural Selection. *Statistics and Computing*, 4(2) :87–112.

[Langlais et al., 2007] Langlais, P., Patry, A., and Gotti, F. (2007). A Greedy Decoder for Phrase-based Statistical Machine Translation. *Proc. of TMI*.

[Langlois, 2015] Langlois, D. (2015). Loria System for the WMT15 Quality Estimation Shared Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 323–329, Lisbon, Portugal. Association for Computational Linguistics.

[Lempa et al., 2010] Lempa, P., Ptaszynski, M., and Masui, F. (2010). A Survey on the Use of Genetic Algorithms in Natural Language Processing. *Advanced Robotics*, 24(5-6) :739761.

[Lepage, 1998] Lepage, Y. (1998). Solving Analogies on Words : an Algorithm. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98/COLING '98, pages 728–734, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Mehrabian and Lucas, 2006] Mehrabian, A. R. and Lucas, C. (2006). A Novel Numerical Optimization Algorithm Inspired from Weed Colonization. *Ecological Informatics*, 1(4) :355–366.

[Menacer et al., 2017] Menacer, M. A., Langlois, D., Mella, O., Fohr, D., Jouvet, D., and Smaïli, K. (2017). Is Statistical Machine Translation Approach Dead ? In *ICNLSSP 2017 - International Conference on Natural Language, Signal and Speech Processing*, pages 1–5, Casablanca, Morocco. ISGA.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent Neural Network Based Language Model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[Nabhan and Rafea, 2005] Nabhan, A. R. and Rafea, A. (2005). Tuning Statistical Machine Translation Parameters Using Perplexity. In *Information Reuse and Integration, Conf, 2005. IRI-2005 IEEE International Conference on.*, pages 338–343. IEEE.

[Nagao, 1984] Nagao, M. (1984). A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, New York, NY, USA. Elsevier North-Holland, Inc.

[Neubig and Watanabe, 2016] Neubig, G. and Watanabe, T. (2016). Optimization for Statistical Machine Translation : Survey. *Computational Linguistics*, pages 1–54.

[Och, 1999] Och, F. J. (1999). An Efficient Method for Determining Bilingual Word Classes. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, pages 71–76. Association for Computational Linguistics.

[Och, 2002] Och, F. J. (2002). Statistical Machine Translation : from Single Word Models to Alignment Templates. In *Thèse de doctorat*.

[Och, 2003] Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

[Och and Ney, 2003] Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Comput. Linguist.*, 29(1) :19–51.

[Otto and Riff, 2004] Otto, E. and Riff, M. C. (2004). Towards an Efficient Evolutionary Decoding Algorithm for Statistical Machine Translation. In *Mexican International Conference on Artificial Intelligence*, pages 438–447. Springer.

[Papadimitriou and Steiglitz, 1998] Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization : Algorithms and Complexity*. Courier Corporation.

[Papineni et al., 2002] 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 on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

[Ross and Corne, 1994] Ross, P. and Corne, D. (1994). Applications of Genetic Algorithms. *AISB Quaterly on Evolutionary Computation*, 89 :23–30.

[Snover et al., 2006] Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, volume 200.

[Stolcke et al., 2011] Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). SRILM at Sixteen : Update and Outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, volume 5.

[Su et al., 1992] Su, K.-Y., Wu, M.-W., and Chang, J.-S. (1992). A New Quantitative Quality Measure for Machine Translation Systems. In *Proceedings of the 14th Conference on Computational Linguistics*, volume 2, pages 433–439. Association for Computational Linguistics.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

[Vauquois and Boitet, 1985] Vauquois, B. and Boitet, C. (1985). Automated Translation at Grenoble University. *Comput. Linguist.*, 11(1) :28–36.

[Willmott and Matsuura, 2005] Willmott, C.-J. and Matsuura, K. (2005). Advantages of the Mean Absolute Error (MAE) Over the Root Mean Square Error (RMSE) in Assessing Average Model Performance. *Climate research*, 30(1) :79–82.

[Zens et al., 2002] Zens, R., Och, F. J., and Ney, H. (2002). Phrase-based Statistical Machine Translation. In *Annual Conference on Artificial Intelligence*, pages 18–32. Springer.

[Zhang et al., 2001] Zhang, L., Madigan, C. F., Moskewicz, M. H., and Malik, S. (2001). Efficient Conflict Driven Learning in a Boolean Satisfiability Solver. In *Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided Design*, pages 279–285. IEEE Press.

[Zogheib, 2011] Zogheib, A. (2011). Genetic Algorithm-based Multi-word Automatic Language Translation. *Recent Advances in Intelligent Information Systems*, pages 751–760.

Résumé

Différentes composantes des systèmes de traduction automatique statistique sont considérées comme des problèmes d'optimisations. En effet, l'apprentissage du modèle de traduction, le décodage et l'optimisation des poids de la fonction log-linéaire sont trois importants problèmes d'optimisation. Savoir définir les bons algorithmes pour les résoudre est l'une des tâches les plus importantes afin de mettre en place un système de traduction performant.

Plusieurs algorithmes d'optimisation sont proposés pour traiter les problèmes d'optimisation du décodeur. Ils sont combinés pour résoudre, d'une part, le problème de décodage qui produit une traduction dans la langue cible d'une phrase source, d'autre part, le problème d'optimisation des poids des scores combinés dans la fonction log-linéaire pour l'évaluation des hypothèses de traduction au cours du décodage. Le système de traduction statistique de référence est basé sur un algorithme de recherche en faisceau pour le décodage, et un algorithme de recherche linéaire pour l'optimisation des poids associés aux scores.

Nous proposons un nouveau système de traduction avec un décodeur entièrement basé sur les algorithmes génétiques. Les algorithmes génétiques sont des algorithmes d'optimisation bio-inspirés qui simulent le processus de l'évolution naturelle des espèces. Ils permettent de manipuler un ensemble de solutions à travers plusieurs itérations pour converger vers des solutions optimales. Ce travail, nous permet d'étudier l'efficacité des algorithmes génétiques pour la traduction automatique statistique. L'originalité de notre proposition est de proposer deux algorithmes : un algorithme génétique, appelé GAMaT, comme décodeur pour un système de traduction statistique à base de segments, et un algorithme génétique, appelé GAWO, pour l'optimisation des poids de la fonction log-linéaire afin de l'utiliser comme fonction *fitness* pour GAMaT. Nous proposons également, une approche neuronale pour définir une nouvelle fonction *fitness* pour GAMaT. Cette approche consiste à utiliser un réseau de neurones pour l'apprentissage d'une fonction qui combine plusieurs scores, évaluant différents aspects d'une hypothèse de traduction, combinés auparavant dans la fonction log-linéaire, et qui prédit le score BLEU de cette hypothèse de traduction.

Dans un premier temps, nous réalisons un ensemble d'expérimentations pour étudier le comportement de nos algorithmes génétiques, GAMaT et GAWO, ainsi que les performances du réseau de neurones pour l'apprentissage de la fonction de prédiction du score BLEU. Par la suite, nous évaluons les performances de traduction de notre système de traduction génétique sur deux paires de langues différentes (français-anglais et turc-anglais). GAMaT est évalué en utilisant les deux fonctions de *fitness*, à savoir GAWO pour l'optimisation des poids et la fonction de prédiction du BLEU apprise par le réseau de neurones. Nous comparons également, les performances de notre système avec celles des systèmes de référence en traduction automatique.

Ce travail, nous a permis de proposer un nouveau système de traduction automatique statistique ayant un décodeur entièrement basé sur des algorithmes génétiques. Les traductions en sortie du système rivalisent avec celles des systèmes de référence, malgré le fait que les performances de traduction sur les ensembles de test ne sont pas meilleures. La thèse propose une

analyse statistique comparative des cas où chaque système donne de meilleurs résultats que les deux autres, où notre système s'est mieux comporté que les autres dans un grand nombre de traductions. Nous avons également étudié les avantages et les limites de l'utilisation des algorithmes génétiques dans un système de TAS, tout en proposant une analyse critique et des perspectives d'amélioration et d'adaptation de notre travail.

Mots-clés: Traduction automatique statistique, Algorithmes génétiques, Algorithmes d'optimisation bio-inspirés.

Abstract

Different components of statistical machine translation systems are considered as optimization problems. Indeed, the learning of the translation model, the decoding and the optimization of the weights of the log-linear function are three important optimization problems. Knowing how to define the right algorithms to solve them is one of the most important tasks in order to build an efficient translation system.

Several optimization algorithms are proposed to deal with decoder optimization problems. They are combined to solve, on the one hand, the decoding problem that produces a translation in the target language for each source sentence, on the other hand, to solve the problem of optimizing the weights of the combined scores in the log-linear function to fix the translation evaluation function during the decoding. The reference system in statistical translation is based on a beam-search algorithm for the decoding, and a line search algorithm for optimizing the weights associated to the scores.

We propose a new statistical translation system with a decoder entirely based on genetic algorithms. Genetic algorithms are bio-inspired optimization algorithms that simulate the natural process of evolution of species. They allow to handle a set of solutions through several iterations to converge towards optimal solutions. This work allows us to study the efficiency of the genetic algorithms for machine translation. The originality of our work is the proposition of two algorithms : a genetic algorithm, called GAMaT, as a decoder for a phrase-based machine translation system, and a second genetic algorithm, called GAWO, for optimizing the weights of the log-linear function in order to use it as a fitness function for GAMaT. We propose also, a neuronal approach to define a new fitness function for GAMaT. This approach consists in using a neural network to learn a function that combines several scores, which evaluate different aspects of a translation hypothesis, previously combined in the log-linear function, and that predicts the BLEU score of this translation hypothesis.

First, we carry out a set of experiments to study the behavior of our genetic algorithms, GAMaT and GAWO, as well as the performances of the neural network to learn the prediction function of the BLEU score. Subsequently, we evaluate the translation performance of our genetic translation system for two pairs of different languages (French-English and Turkish-English). GAMaT is evaluated using the two fitness functions, namely GAWO for optimizing weights and the BLEU prediction function learned by the neural network. We compare also the performance of our system with those of reference systems in machine translation.

This work allowed us to propose a new machine translation system with a decoder entirely based on genetic algorithms. The output translations of the system compete with those of the reference systems, despite the fact that the translation performance on the test sets is not better. The thesis proposes a comparative statistical analysis of the cases where each system gives better results than the two others, where our system has gotten done better than the other systems in a large number of translations. We also studied the benefits and limitations of using genetic algorithms in a machine translation system, while providing critical analysis and opportunities for improvement and adaptation of our work.

Keywords: Statistical Machine Translation, Genetic Algorithms, Bio-inspired Optimization Al-

gorithms.