# CHAPITRE III
# Méthodes de recherche des *N* meilleures solutions

## 1 Introduction

Les méthodes de recherche des *N* meilleures solutions sont essentiellement apparues, en reconnaissance automatique de la parole, vers la fin des années 80 et le début des années 90. Le critère de validité de chaque méthode est l'optimalité.

**Une méthode est optimale si les *N* solutions fournies sont les *N* meilleures (i.e. s'il n'y a pas d'oubli).**

Les *N* solutions obtenues sont des solutions syntaxiquement différentes et non pas des alignements différents d'une même solution.

Nous allons décrire dans ce qui suit une rétrospective générale sur les méthodes de recherche des *N* meilleures solutions.

## 2 Méthodes de recherche

### 2.1 A partir de la solution optimale

La phase de recherche des *N* meilleures solutions se déroule en même temps que le processus de décodage. [Myers, 81-b] et [Lee, 89] ont proposé la même idée, mais le calcul était différent. La recherche des *N* meilleures solutions se déroulait au niveau syntaxique. Ils se sont intéressés uniquement à la recherche de la seconde solution. A chaque état syntaxique du réseau, ils mémorisaient les deux chemins possédant les scores les plus élevés, et ils ne prolongeaient par la suite que **le meilleur des deux**. De ce fait l'optimalité n'est pas garantie. La figure suivante (III.1) illustre cette recherche.

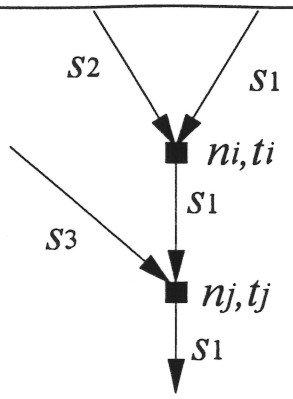


Figure III.1 Progression de recherche par Myers

Deux meilleurs chemins $s_1$ et $s_2$ se croisent au noeud $n_i$ et à l'instant $t_i$, les deux sont mémorisés cependant seul le chemin $s_1$ est prolongé (en supposant que c'est le meilleur pour ce noeud). A un autre instant $t_j$, $(j \neq i)$ et au noeud $n_j$, $s_1$ se croise avec $s_3$, les deux sont mémorisés et $s_1$ est prolongé, ainsi de suite .... La non-optimalité provient du traitement effectué au niveau du noeud $n_j$.

En effet, on ne sait pas si le score affecté au chemin $s_2$ (après prolongement) aurait été supérieur ou non au score affecté au chemin $s_3$. La meilleure solution correspond à la solution optimale obtenue par l'algorithme de Viterbi. La seconde meilleure solution est obtenue en prolongeant dans chaque noeud syntaxique le second meilleur chemin mémorisé.

Le résultat obtenu est sous-optimal car la seconde solution n'est qu'une combinaison d'une séquence partielle de la meilleure solution et une séquence du second meilleur chemin. On a donc une seconde solution mais elle est parfois incorrecte.

## 2.2 A partir de l'algorithme One-Pass

[Steinbiss, 89] proposa une méthode exacte de recherche des $N$ meilleures solutions et cela à partir de l'algorithme One-Pass. A chaque noeud et pour chaque instant, on mémorise dans une liste les $N$ meilleurs scores correspondant à des séquences de mots différentes. La figure (III.2) présente la procédure de recherche des 2 meilleures solutions. L'extension à $N$ solutions ne pose pas de problème.

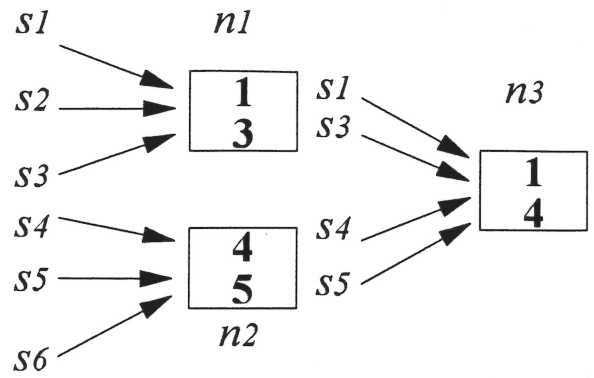


Figure III.2 Progression optimale par Steinbiss

Au noeud $n_1$ arrivent trois chemins ($s_1$, $s_2$, $s_3$), on choisit les deux meilleurs qui fournissent les scores à mémoriser ($s_1$ et $s_3$, par exemple). La même chose est faite pour le noeud $n_2$. On récupère ainsi $s_4$ et $s_5$. De ces quatre chemins, le noeud $n_3$ va sélectionner les deux meilleurs ($s_1$ et $s_4$) ainsi de suite. A la fin, on remonte les deux chemins pour avoir les séquences complètes (dans ce cas les séquences correspondantes aux chemins $s_1$ et $s_4$).

On retrouve pratiquement la même idée dans [Mariño, 89]. La génération des $N$ meilleures solutions a été réalisée sur des unités phonétiques connectées à partir de l'algorithme One-Pass.

La mise en oeuvre d'une telle procédure nécessite de mémoriser pour chaque noeud acoustique $N$ solutions partielles (score et séquence de mots reconnus ; voir figure III.2). La sélection des $N$ meilleurs chemins s'effectue par un algorithme de tri. La procédure de recherche risque d'être coûteuse si on est amené à trier à chaque instant pour décider des chemins à développer. C'est pour cela que [Steinbiss, 89] proposa une solution de rechange, moins coûteuse en calcul, pour laquelle le résultat obtenu était sous-optimal. Les $N$ meilleures solutions sont obtenues à partir de la segmentation de la meilleure solution. De ce fait chaque solution contient une séquence de mots (un ou plusieurs mots) de la solution optimale. La figure suivante (III.3) illustre un exemple de cette procédure et met en évidence la sous-optimalité possible.

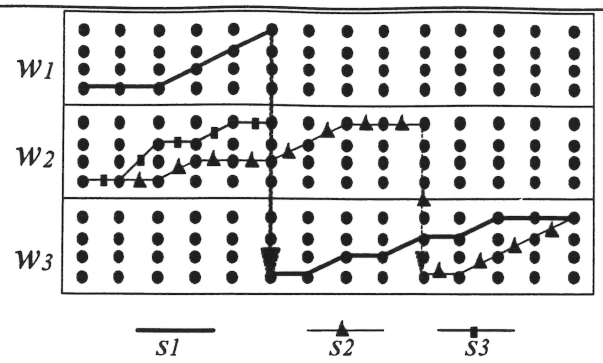


Figure III.3 Progression sous optimale par Steinbiss

$s_1$ correspond au chemin optimal, $s_2$ est le second meilleur chemin. Cependant lors de la procédure de reconnaissance $s_2$ sera éliminé au profit de $s_1$ et il sera donc perdu. A l'instant où $s_1$ passe par une règle inter-unités, on regarde au niveau du noeud syntaxique quel est le mot possédant le second meilleur score de reconnaissance. Dans le cas de la figure (III.3), c'est le mot $w_2$. Ainsi, on a construit un nouveau chemin $s_3$ véhiculant une solution sous optimale. La seconde meilleure solution (fournie par l'algorithme) est obtenue en combinant le chemin $s_3$ avec le reste du chemin optimal $s_1$. Cette dernière méthode rejoint l'idée développée dans le paragraphe 2.1.

## 2.3 A partir d'une méthode de recherche en largeur

Dans la même optique, [Schwartz, 90] a proposé d'agir cette fois-ci sur l'état syntaxique au lieu de l'état acoustique. Sur chaque noeud du réseau syntaxique est définie une liste. Cette dernière sert à mémoriser les $N$ meilleures séquences de mots arrivant sur ce noeud et cela pour chaque instant. Si deux chemins correspondent à la même séquence de mots arrivant à un même noeud à un instant donné, leurs probabilités sont sommées. Les $N$ meilleures séquences de mots sont propagées sur la suite du réseau pour les trames suivantes. Ce choix est guidé par une heuristique. On ne garde dans la liste de chaque noeud syntaxique que les solutions ayant un score supérieur à un seuil fixé au départ par rapport à la meilleure solution contenue dans cette même liste. Ce type de recherche correspond à une méthode de Viterbi avec progression en largeur. La complexité de calcul au niveau de chaque noeud syntaxique est de l'ordre de $O(N^2)$ et se ramène à $O(\sqrt{N})$ [Schwartz, 90] en utilisant des techniques de "pruning". Une technique de "pruning" consiste à effectuer une sélection des chemins à l'intérieur de la pile affectée à chaque noeud du réseau syntaxique.

A partir de cette méthode, deux techniques sous-optimales ont été introduites pour réduire le temps de calcul qui reste malgré tout très élevé. On retrouve ces deux méthodes dans [Schwartz, 91].

- La première consiste à garder dans chaque noeud syntaxique les $N$ meilleurs chemins arrivant à ce noeud, puis à propager la meilleure solution (celle possédant le meilleur score) sur le reste du réseau (i.e. poursuivre le reste du décodage par la procédure classique). A la fin, on remonte récursivement à partir du dernier noeud pour retrouver les $N$ meilleures solutions. Ces solutions sont obtenues d'une manière indirecte à partir de la meilleure solution. Cette idée rejoint celle proposée dans [Steinbiss, 89] et décrite par la figure (III.3) précédente. Cependant elle reste une méthode sous-optimale très peu performante.

- La seconde dépend du mot obtenu à l'intérieur des $N$ meilleures solutions. On mémorise dans chaque noeud syntaxique les $n$ ($n \langle\langle N$)[1] meilleures séquences de mots. Ces séquences seront propagées par la suite sur le reste du réseau. Dans ce même article, les auteurs décrivent un backtracking à l'intérieur de la liste à partir d'une méthode de recherche en profondeur pour retrouver les $N$ meilleures solutions.

Les tests réalisés par [Schwartz, 91] montrent que lorsque le nombre $N$ de solutions demandées augmente, les meilleurs résultats sont obtenus par la méthode, sous-optimale, dépendant du mot obtenu à l'intérieur des $N$ meilleures solutions. La stratégie finale adoptée [Bates, 92] est de rechercher les $N$ meilleures solutions en deux phases.

- Une phase aller en utilisant une grammaire bigramme et des modèles de Markov discrets pour mémoriser les scores de chaque mot rencontré aux différents instants [Austin, 91].

- Une phase retour produit la liste des $N$ meilleures solutions en utilisant l'algorithme des $N$ meilleures solutions dépendant du mot "Word-Dependent Nbest" [Schwartz, 91].

Par ailleurs l'algorithme des $N$ meilleures solutions dépendant du mot a été également employé dans une application ayant un réseau acoustique sous forme d'un arbre et un modèle de langage [Steinbiss, 92]. Un modèle de langage unigramme est utilisé pour générer les $N$ meilleures solutions et un autre modèle de langage bigramme sert pour retrouver les 100 meilleures solutions dans la liste proposée.

## 2.4 A partir de l'algorithme A*

Dans [Soong, 91] les $N$ meilleures solutions sont générées par une méthode Forward-Backward (aller-retour) et cela en deux passes. La première passe, effectuée de manière synchrone, sert à calculer, par l'algorithme de Viterbi, et à mémoriser les différentes probabilités d'émission (meilleur chemin) à chaque instant et pour chaque état du réseau syntaxique. La phase retour, réalisée de manière asynchrone, utilise l'algorithme A* pour

---

[1]Dans [Steinbiss, 92], lorsque $N = 50$, les performances ne varient pas quand $n$ passe de 3 à 5.

retrouver les $N$ meilleures séquences de mots (les $N$ meilleurs chemins distincts possibles entre le noeud final et le noeud de départ). L'estimation de la portion du signal non encore traitée provient des valeurs mémorisées pendant la phase aller. Cette estimation est optimale et conduit alors à une recherche très efficace des $N$ meilleures solutions. Dans cette approche le temps de calcul croît avec le nombre de solutions développées. Ce temps devient critique pour un $N$ très élevé. Cette méthode a été utilisée dans les systèmes de reconnaissance VOYAGER [Zue, 90] et CRIM [Cardin, 92].

Dans [Lokbani, 92], la méthode de Soong est utilisée pour générer les $N$ meilleures solutions à partir d'une adaptation du logiciel PHIL90. Cependant nous développons les $N$ meilleures solutions pour obtenir le score, l'alignement (chemin) et l'étiquette de chaque solution développée (informations utilisées dans la suite). Toutes ces données sont mémorisées pendant la phase retour dans une pile. Ainsi, on n'agit plus au niveau syntaxique mais au niveau acoustique. De ce fait les données mémorisées dans la pile sont plus importantes. Afin d'avoir un temps de recherche efficace, il faut gérer correctement la manière de développer et de mémoriser les différentes données des différents noeuds traités. Le chapitre suivant va décrire notre façon de procéder afin d'avoir une recherche rapide des $N$ meilleures solutions tenant compte des informations nécessaires pour la suite de nos travaux.

## 2.5 A partir des $N$ meilleurs scores

Enfin, on clôt cette rétrospective par une méthode simple et optimale, dans le cas des mots isolés. Les modèles de Markov permettent de calculer pour chaque mot du vocabulaire la probabilité d'émission de l'observation $X$ connaissant le modèle. A la fin, on détermine les $N$ meilleurs modèles ayant les meilleurs scores. Si cette méthode s'applique bien dans le cas d'un petit vocabulaire (une dizaine de mots) et pour une application de recherche des $N$ meilleures solutions sur des mots isolés, son extension dans le cas d'un vocabulaire élevé ou pour une application de mots connectés est hors prix (mémoire, temps de calcul, ...).

Un deuxième problème posé par cette méthode est lié au partage des silences de début et de fin d'enregistrement par les différents modèles. Les différentes configurations possibles sont représentées sur la figure III.4

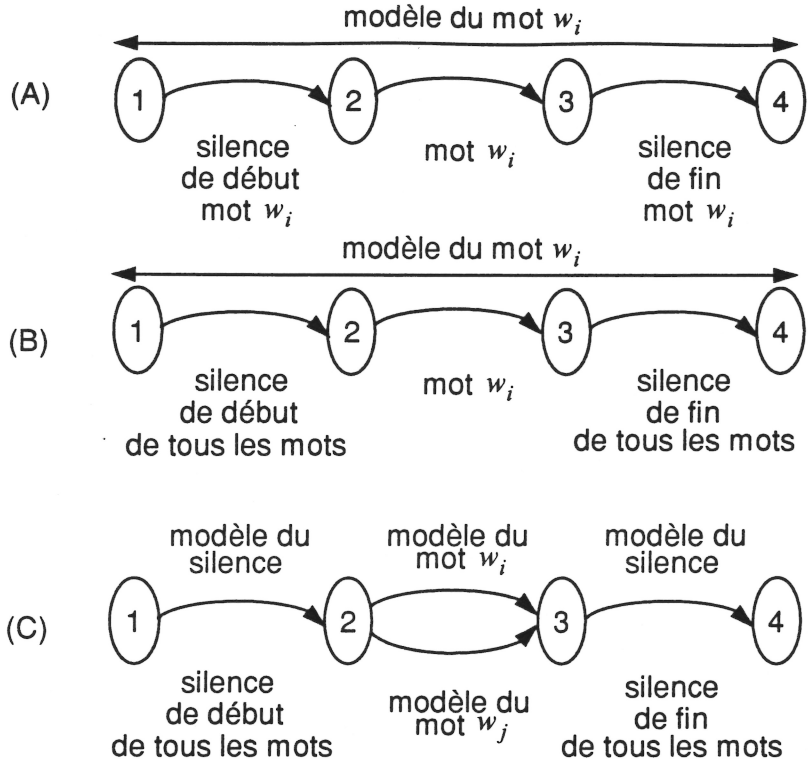


Figure III.4 Configurations de recherche des $N$ meilleures solutions

Ne pas partager les unités du silence implique que le modèle associé à chaque mot va avoir un apprentissage trop sélectif, de ce fait ce modèle risque de mal se généraliser sur le test. Les frontières entre les silences de début et de fin d'enregistrement et le mot deviennent des frontières apprises localement.

Pour éviter ces problèmes d'apprentissage, il serait intéressant de partager les modèles de silence entre tous les mots du vocabulaire (configuration B sur la figure III.4). Cependant lors de la recherche des $N$ meilleures solutions, tous les modèles (de silence) seront dupliqués autant de fois qu'il y a de mots dans le vocabulaire.

De ce fait, il serait intéressant d'utiliser la configuration (C) sur la figure III.4. La recherche des $N$ meilleures solutions pour des mots isolés devient une recherche pour des mots connectés puisque chaque solution va contenir {silence de début, un mot et silence de fin}. Les méthodes optimales de recherche des $N$ meilleures solutions citées précédemment peuvent être utilisées dans ce cas.