Des implémentations parallèles pour une application de la RAP*

Y. O. Mohamed El Hadj

Université Paul Sabatier, IRIT, 118 Route de Narbonne, 31062 Toulouse, France. Email : elhadj@irit.fr

* *Ce travail est financé par l'AUPELF-UREF. Il a été initié au Laboratoire LaRI, Oujda-Maroc, sous la direction de E. M. DAOUDI et se poursuit actuellement à l'IRIT, Toulouse-France, sous l'encadrement de R. A. OBRECHT.*

Résumé

We show through this work that harnessing the power of parallel machines can increase greatly the speed and storage capacity of certain recognizers. Obtained improvements can be exploited to expand the vocabulary on existing real-time tasks or to increase the modelization precision where the recognition accuracy is most interesting that the real time.

1. Introduction

La Reconnaissance Automatique de la Parole (RAP) a pour ambition de réaliser une communication homme-machine orale avec le maximum de convivialité. De très nombreux travaux ont été effectués et l'effort de recherche croît de jour en jour. Cependant, malgré les intéressants progrès, la réalisation d'un dialogue homme-machine dans un langage naturel reste toujours un rêve pour le public et un défi pour les spécialistes. La principale raison de cet état de fait est qu'une communication orale homme-machine sans contrainte nécessite la mise en œuvre de nombreuses connaissances difficiles et coûteuses à rassembler et à formaliser.

Trois critères de base peuvent différer les objectifs d'applications de la RAP. Il s'agit de la taille du vocabulaire, qui peut être petit, moyen ou grand, le fait que les mots soient prononcés de manière continue ou isolée et enfin le nombre de locuteurs destinés à utiliser le système. On distingue ainsi :

- **les systèmes à commande vocale :** qui sont généralement caractérisés par des vocabulaires de taille petite à moyenne, par l'indépendance du locuteur, de l'environnement et par un taux de reconnaissance élevé,
- **les systèmes de dictée vocale :** qui, à l'inverse des systèmes précédents, peuvent avoir un vocabulaire de taille très élevée et une syntaxe naturelle ; la précision de la reconnaissance est cependant moins imposée car les erreurs peuvent être corrigées par la suite sans poser de vrais problèmes,
- **les systèmes de compréhension :** qui sont amenés à percevoir la signification du message et de réagir, soit par l'émission d'une réponse vocale, soit par une action mécanique sur l'environnement.

Dans ce travail, nous nous intéressons à la parallélisation d'une application de la reconnaissance de mots isolés multilocuteurs. Ce type d'applications est en forte expansion puisqu'il répond aux besoins d'une utilisation réelle que l'on rencontre dans différents domaines. Une modélisation très fine permettant de réduire le risque d'ambiguïté et de différer au mieux les mots du vocabulaire est donc nécessaire pour aboutir à cette fin. Ceci n'est généralement pas sans prix étant donné que le coût du traitement augmente considérablement. Nous commençons par présenter la structure du modèle séquentiel de l'application étudiée. Nous mettons l'accent sur la construction du réseau global et le modèle utilisé pour l'application. Il s'agit du modèle à deux niveaux centiseconde développé par M. Meziane [4] qui sera noté cTLHMM (pour *centisecond Two Level Hidden Markov Model* en anglais). Les modèles de Markov cachés (HMM en anglais) sont largement utilisés dans la reconnaissance de la parole grâce à leur efficacité établie. Cependant, ils ne tiennent pas compte, de manière satisfaisante, de la durée des sons (prosodie) ; cTLHMMs ont donc été introduits pour manipuler les aspects prosodiques de la parole. Nous rappelons ensuite deux stratégies de parallélisation déjà étudiées et nous signalons la nécessite d'équilibrer la charge des processeurs. Plusieurs techniques sont alors proposées pour résoudre ce problème et des expérimentations sont menées.

2. Modèle séquentiel

Pour une application de mots isolés avec petit vocabulaire, les mots sont souvent modélisés globalement par des HMMs séparés. Ces modèles sont appris, chacun sur un corpus d'apprentissage formé par suffisamment de prononciations du mot associé. En mode reconnaissance, la vraisemblance d'une suite d'observations est calculée par rapport à chaque modèle. Le mot dont le modèle fournit la meilleure vraisemblance est considéré comme mot reconnu. Cette façon de faire est évidemment la plus simple, mais elle devient impossible quand le vocabulaire augmente et/ou lorsqu'il s'agit de la reconnaissance multilocuteurs. En effet, il y a non seulement le besoin d'avoir un corpus d'apprentissage très large pour chaque mot du vocabulaire, mais aussi le risque de similarité entre les mots. Dans ce dernier cas, il sera difficile de distinguer des mots acoustiquement proches comme par exemple *complément* et *compliment*. La distinction entre eux ne peut se faire que si leurs modèles ont été construits à partir de modèles élémentaires de leurs phonèmes. De plus, si de nouveaux mots sont ajoutés au vocabulaire, leurs modèles globaux devront être appris séparément nécessitant ainsi un certain nombre de prononciations de chacun d'entre eux. Ceci est infaisable pour un vocabulaire large et un système multilocuteurs.

Une approche pratique consiste à construire le modèle de mot à l'aide de la concaténation des modèles de petites unités partagées par les mots du vocabulaire. Ces unités sont en principe moins nombreuses par rapport aux mots du vocabulaire. Leurs modèles sont petits avec peu de paramètres à estimer. Ces unités peuvent être de nature linguistique comme, des phonèmes, des triphones, des allophones ou encore des segments qui prennent en considération des caractéristiques articulatoires, etc. D'autres unités de nature acoustique existent, comme le *fénone* chez IBM. Dans ce travail, nous avons choisi l'unité *pseudo-diphone* qui modélise à la fois les parties stables des phonèmes et les parties transitoires entre phonèmes. Le modèle global de l'application est ensuite construit hiérarchiquement par concaténation de modèles élémentaires d'unités choisies comme unités de base. En fusionnant toutes les connaissances et en reliant les modèles de mots par une entrée et une sortie communes nous obtenons le modèle global de l'application (HMM standard ou cTLHMM). L'entrée et la sortie sont modélisées par des HMMs représentant respectivement le silence de début et le silence de fin des mots.

De manière succincte, un HMM est un graphe probabilisé défini par, son ensemble d'états acoustiques $Q = \{q_1, q_2, \ldots, q_N\}$, sa distribution des probabilités initiales $\Pi = (\pi_i)_{1 \leq i \leq N}$, sa matrice des probabilités de transitions $A = (a_{ij})_{1 \leq i,j \leq N}$ et ses lois de probabilités associées aux transitions $B = (b_{ij}(\cdot))_{1 \leq i,j \leq N}$. Quant au cTLHMM, les paramètres sont à peu près les mêmes sauf qu'il y a des lois de probabilités qui gèrent le temps de séjour au niveau des unités phonétiques : $\Delta = (\rho_k(\cdot))_{1 \leq k \leq K}$, où K désigne le nombre d'unités phonétiques distinctes à modéliser. Pour réduire le nombre total de paramètres du modèle global, noté $\lambda = (\Pi, A, B)$ pour les HMMs et $\lambda = (\Pi, A, B, \Delta)$ pour les cTLHMMs, toutes les occurrences d'une unité phonétique font référence à un seul ensemble des lois de probabilités. Contrairement aux lois et dans le cas d'un vocabulaire de faible taille comme dans notre catégorie d'applications, les états acoustiques d'une unité phonétique sont dupliqués autant de fois que cette unité apparaît dans le réseau. Les paramètres du modèle obtenu sont estimés sur un ensemble d'apprentissage à l'aide d'une procédure de réestimation itérative qui consiste à déterminer, à partir d'un modèle initial λ_0, un nouveau modèle λ_1 pour lequel la vraisemblance des observations est maximale et à répéter la procédure jusqu'à satisfaction d'une condition d'arrêt. Les formules de réestimation obtenues à chaque itération ne tiennent compte que des informations apparues le long des chemins les plus probables. Ceci revient à faire des statistiques sur les événements apparus le long de ces chemins, dits optimaux.

3. Étude de la parallélisation

Nous considérons pour cette étude une architecture à mémoire distribuée composée de p processeurs, notés $(P_i)_{0 \leq i \leq p-1}$. Chaque processeur possède sa propre mémoire et communique avec les autres via un réseau d'interconnexion. Nous désignons par m le nombre de mots du vocabulaire de l'application traitée.

3.1. Besoin de parallélisation

Les modèles markoviens ont fait preuve de leur efficacité dans le domaine de la RAP. Toutefois, leur mise en œuvre nécessite un volume de données assez important pour apprendre correctement leurs paramètres. Ceci peut conduire à une phase d'apprentissage longue et fastidieuse, non seulement pour des vocabulaires de grande taille, mais aussi pour des petits vocabulaires où les mots sont acoustiquement proches. De plus l'apprentissage ne se fait pas toujours une seule fois. Il peut être repris à chaque nouvel environnement et parfois à chaque nouvel utilisateur. Il est donc intéressant de pouvoir faire un apprentissage rapide afin d'accroître l'adaptabilité du système et d'encourager son utilisation. Le besoin de faire la parallélisation se fait alors sentir pour l'apprentissage. De même, si la parallélisation est intéressante pour une phase d'apprentissage, elle l'est également pour la reconnaissance. En effet, l'obtention d'un temps de reconnaissance raisonnable est toujours un rêve qui préoccupe les concepteurs des systèmes de RAP. D'ailleurs, pour fournir assez rapidement une réponse, les systèmes de reconnaissance font toujours le compromis entre la taille du vocabulaire et la précision de la modélisation, bien que ceci ait une répercussion directe sur le taux de reconnaissance.

3.2. Quelques parallélisations existantes

Les tentatives de parallélisation existantes se focalisent principalement sur l'algorithme qui détermine les chemins optimaux (Viterbi pour les HMMs standard) afin de cerner des parties de calcul indépendantes. Pour une prononciation de T observations et un modèle de N états, cet algorithme s'écrit :

Pour $t = 1$ *à* T *faire : Pour* $j = 1$ *à* N *faire :* $\delta(t, q_j) = \max_{q_i} \Big(\delta(t-1, q_i) \times a_{ij} \times b_{ij}(y_t) \Big)$.

La difficulté de parallélisation de cet algorithme vient du fait que les observations sont traitées de manière séquentielle : l'observation y_t ne peut être traitée qu'après la fin du traitement de l'observation y_{t-1}. Il en résulte une forte dépendance dans la boucle t. Afin de briser cette dépendance, nous trouvons dans l'article [6] une transformation de cette boucle, mais le taux de la reconnaissance est malheureusement dégradé. D'autres tentatives de parallélisation au niveau de la boucle j sont proposées dans les articles [7, 2]. Cependant, les performances sur une machine à mémoire distribuée sont modestes à cause du volume de communications induit par la distribution des états.

3.3. Parallélisation de l'apprentissage

Notre stratégie de parallélisation pour la phase d'apprentissage consiste à traiter les mots en parallèle en distribuant équitablement le vocabulaire aux processeurs. Pour minimiser les échanges entre les processeurs, le réseau global de l'application est copié dans la mémoire de chaque processeur. On l'appelle *stratégie de duplication*. Contrairement au réseau, l'ensemble d'apprentissage est distribué aux processeurs. Chaque processeur reçoit donc un corpus d'apprentissage local formé par les prononciations des $\frac{m}{p}$ mots dont il est responsable.

Au début de chaque itération d'apprentissage, les processeurs travaillent simultanément de manière indépendante pour déterminer les chemins optimaux associés aux prononciations de leurs propres mots. À la fin de l'itération, les processeurs doivent réestimer le réseau global. Or, la réestimation de ce réseau nécessite des statistiques globales sur l'utilisation des paramètres. Chaque processeur doit donc déterminer les informations relatives à l'utilisation des états, des transitions et des lois le long de ses propres chemins optimaux. Ces informations sont d'abord traitées localement avant d'être échangées entre les processeurs afin de diminuer le volume de données à communiquer. Pour minimiser le coût de la communication, nous avons essayé de la recouvrir par la réestimation des probabilités de la partie locale du réseau. C'est-à-dire que pendant cette communication, un processeur peut anticiper la réestimation des probabilités des transitions relatives aux modèles de ses propres mots.

Cette stratégie a l'avantage d'être facile à étudier et à mettre en œuvre. Cependant, elle conserve le même espace d'exploration sur tous les processeurs à cause de la duplication du réseau global. Donc la reconnaissance parallèle ne peut être utile que dans le cas où plusieurs mots seraient à reconnaître simultanément.

3.4. Parallélisation de l'apprentissage et de la reconnaissance

Nous adoptons une nouvelle stratégie qui distribue aux processeurs le corpus et le réseau. Chaque processeur reçoit un vocabulaire local issu d'une répartition toujours équitable du vocabulaire. Il construit alors son propre réseau à partir des modèles de ces mots. Le réseau local ainsi obtenu possède les mêmes caractéristiques que le réseau global, à savoir la structure hiérarchique et la présence des modèles de silences. Ces derniers sont dupliqués sur tous les processeurs afin de pouvoir faire la reconnaissance parallèle et de garder l'analogie avec le cas séquentiel.

Cette stratégie que nous appellons *stratégie de distribution* a l'avantage de bien gérer la mémoire globale de la machine parallèle et de réduire l'espace de travail de chaque processeur en lui affectant seulement la partie du réseau associée à ses propres mots.

Le réseau local de chaque processeur est appris sur un corpus d'apprentissage formé par les différentes prononciations des $\frac{m}{p}$ mots qui lui sont associés. D'une itération à l'autre, les processeurs doivent réestimer leurs réseaux locaux. Ceci nécessite l'échange des informations relatives à l'utilisation des paramètres communs aux processeurs, à savoir ceux des modèles des silences ainsi que ceux relatifs aux lois de probabilités partagées par les différentes occurrences d'une même unité phonétique et qui peuvent se trouver sur des processeurs différents. Comme il est coûteux, au niveau temps de contrôle, de gérer cette dépendance, nous avons légèrement modifié la modélisation. Les unités phonétiques de chaque mot sont alors modélisées indépendamment de celles des autres mots. Les occurrences d'une unité phonétique d'un même mot partagent toujours le même ensemble de lois. Comme toute modélisation dépendante du contexte, le nombre de paramètres du modèle global augmentent et le volume des données nécessaire à l'apprentissage s'élargit puisqu'il n'y a plus de partage de certaines connaissances globales. Toutefois, sur une machine parallèle, un niveau de complexité élevé peut être accepté dans la mesure où il est compensé par les avantages offerts par cette machine. Pour connaître la répercussion de cette nouvelle modélisation, nous avons calculé des taux de reconnaissance sur un vocabulaire de 20 mots après un apprentissage sur le même corpus. Nous n'avons pas remarqué une dégradation de la qualité de reconnaissance, mais plutôt une amélioration de 3.5 %, ce qui justifie son adéquation. Avec cette modélisation, la réestimation du réseau local nécessite seulement l'échange des résultats des statistiques relatives aux modèles de silences. Durant cette communication un processeur peut d'ores et déjà entamer la réestimation des modèles de mots qui lui sont associés. Seuls, les modèles de silences sont obligés d'attendre la fin de la communication pour être réestimés.

Au niveau reconnaissance, des recherches simultanées des chemins optimaux dans les réseaux locaux sont faites. Le mot associé au chemin optimal local qui a donné la meilleure probabilité est considéré comme mot reconnu. Pour ce faire, il suffit de faire circuler les meilleures probabilités locales entre les processeurs pour en choisir la plus grande. Ensuite, une décision de reconnaissance sera prise par le processeur qui possède cette plus grande probabilité. Avec cette façon de faire, le coût de la reconnaissance ne dépend plus de la taille du réseau global, mais plutôt de celle d'un réseau local de $\frac{m}{p}$ mots. Signalons que le problème de partage des lois ne se pose pas pour la reconnaissance puisqu'il ne s'agit pas de faire des réestimations. Donc la stratégie de distribution peut être directement utilisée pour la reconnaissance sans modifier la modélisation habituelle comme pour l'apprentissage.

4. Équilibrage de charges

Les stratégies de parallélisation précédantes ont été implémentées dans les environnements PVM (*Parallel Virtual Machine*) et MPI (*Message Passing Interface*) et testées sur deux sortes de machines parallèles : TN310 (32 Transputers T9000) et PoPC (cluster de 12 PCs interconnectés par le réseau rapide myrinet). A travers ces travaux [1, 5], nous avons remarqué un fort déséquilibre entre la charge des processeurs. À l'aide d'une étude de la complexité théorique du calcul corroborée par des résultats expérimentaux, nous avons montré que le déséquilibre provient de la mauvaise répartition du calcul des chemins optimaux et donc de la façon dont les mots sont distribués. D'où la nécessité de chercher une distribution qui se base sur la complexité du calcul relative à chaque mot.

Dans le but d'obtenir une bonne répartition de charge, plusieurs heuristiques sont essayées. Nous commençons par trier les mots par ordre de coût décroissant dans un vecteur C. En nous basant sur ce vecteur, nous affectons ensuite les mots aux processeurs selon différentes stratégies qui dérivent toutes de la technique LPTF (*Largest Processing Time First*) [3].

4.1. *Distribution cyclique*

Nous affectons les mots aux processeurs de manière cyclique selon leur coût d'apprentissage. Le mot dont le coût se trouve dans la $i^{\text{ème}}$ composante du vecteur C sera affecté au processeur $P_{(i \ mod \ p)}$. Cependant, cette stratégie a tendance à surcharger les premiers processeurs par rapport aux derniers. En effet, au cours de chaque passage, ce sont les premiers processeurs qui reçoivent les mots les plus coûteux parmi ceux qui n'ont pas encore été distribués.

4.2. *Distribution par permutation*

Au lieu d'utiliser une distribution cyclique simple, nous utilisons plutôt une permutation sur les processeurs. Au premier passage, les mots sont affectés aux processeurs dans l'ordre $P_0,\ P_1$, jusqu'à P_{p-1}. Au deuxième passage, nous utilisons une permutation sur les processeurs, c'est-à-dire l'ordre $P_{p-1},\ P_0,\ P_1,\ P_2$, jusqu'à P_{p-2} et ainsi de suite. Le mot correspondant à la $i^{\text{ème}}$ composante du vecteur des coûts C sera affecté au processeur $P_{((i-\frac{i}{p}) \ mod \ p)}$.

4.3. *Distribution par alternance de sens*

Au lieu d'utiliser une permutation sur les processeurs, nous revenons sur la distribution cyclique mais en utilisant à chaque nouveau passage un ordre inverse sur les processeurs (parcours gauche $\rightarrow$ droite puis droite $\rightarrow$ gauche (GD-DG)). Au premier passage, nous distribuons les mots sur les processeurs dans l'ordre $P_0,\ P_1$, jusqu'à P_{p-1}. Lors du deuxième passage, nous utilisons carrément l'ordre inverse, c'est-à-dire $P_{p-1},\ P_{p-2}$, jusqu'à P_0 et ainsi de suite.

4.4. *Expérimentations*

Les distributions que nous venons de voir sont intégrées dans les programmes PVM à base de HMMs et cTLHMMs. Les expérimentations correspondantes sont faites sur la machine TN310. À cause du faible capacité de cette machine, nous n'avons pas pu aller au-delà de 20 mots pour les HMMs et 10 mots pour les cTLHMMs. Chaque mot est prononcé par chacun des 6 locuteurs pour former le corpus d'apprentissage et une fois pour former l'ensemble test. Dans le tableau suivant, nous comparons les mesures de performances (temps d'exécution moyens en secondes sur p processeurs T_p et efficacités E_p ($E_p = \frac{T_1}{pT_p}$)) d'une itération de l'apprentissage relativement aux différentes distributions. Nous constatons que les performances sont largement améliorées par rapport à la distribution initiale qui affecte les mots aux processeurs indépendamment de leurs coûts. Cette amélioration est plus intéressante pour les deux dernières distributions. En effet, les efficacités les plus modestes sont passées de 41.57% (resp. 58.25%) à 51.13% (resp. 67.84%) pour la duplication (resp. distribution) avec les HMMs sur 10 processeurs.

5. Conclusion

Nous avons montré l'importance de l'utilisation de la complexité théorique de calcul pour obtenir une bonne répartition de charge sur les processeurs. À travers les différentes stratégies de distributions proposées, nous avons remarqué que les performances sont largement améliorées par rapport à la distribution initiale. Cette dernière affectait les mots aux processeurs indépendamment de leur coût. Cependant, le déséquilibre de charge n'a pas complètement disparu. Ceci peut être dû à la distribution équitable qui limite la possibilité d'ajustement de charges à cause de la présence de $\frac{m}{p}$ mots sur chaque processeur et de la petite taille de notre vocabulaire qui ne permet pas d'avoir tous les types de mots. Nous pensons qu'il est encore possible de réduire le déséquilibre et donc améliorer le résultat en utilisant un vocabulaire plus large et une distribution qui n'affecte pas automatiquement $\frac{m}{p}$ mots à chaque processeur, mais plutôt un nombre variable de telle sorte que les charges globales soient approximativement égales.

Références

[1] E. M. Daoudi, A. Meziane, Y. O. Mohamed El Hadj, *Study of Parallelization of the Training for Automatic Speech Recognition*, HPCN'2000, LNCS, 2000, vol. 1823, pages 576-579.

[2] M. Fleury, A. C. Downton, A. F. Clark, *Parallel Structure in an Integrated Speech-Recognition Network*, Europar'99, PP.995-1004, 1999.

[3] R. -L. Graham, *Bounds on Multiprocessing Timing Anomalies*, SIAM J. Appel. Math., 17, PP. 416-429, 1969.

[4] A. Meziane, *Introduction de la durée dans un HMM au niveau supra segmental*, Doctorat d'état, Univ. Oujda, 1997.

[5] Y. O. Mohamed El Hadj, N. Revol, *Parallelization of Automatic Speech Recognition*, soumis pour publication au journal IEEE/SAP.

[6] H. Noda, M. N. Shirazi, B. Zhang, *A parallel processing algorithm for speech recognition using MRF*, IEIC, 4, PP. 819-826, 1993.

[7] S. Phillips, A. Rogers, *Parallel Speech Recognition*, Eurospeech'97, Vol. 1, PP. 135-138, 1997.

Distribution	Stratégie de duplication							Stratégie de distribution						
	$p=1$	$p=2$		$p=5$		$p=10$		$p=1$	$p=2$		$p=5$		$p=10$	
	T_1	T_2	E_2	T_5	E_5	T_{10}	E_{10}	T_1	T_2	E_2	T_5	E_5	T_{10}	E_{10}
HMM														
initiale	20.91	19.24	54.34	9.04	46.26	5.03	41.57	22.25	17.96	61.94	7.40	60.14	3.82	58.25
cyclique	20.91	17.91	58.37	7.47	55.98	4.47	46.78	22.25	15.56	71.50	6.40	69.53	3.35	66.42
perm.	20.91	17.25	60.61	7.24	57.76	4.09	51.13	22.25	15.53	71.64	6.28	70.86	3.28	67.84
GD-DG	20.91	17.19	60.82	7.29	57.37	4.09	51.13	22.25	15.50	71.77	6.38	69.75	3.28	67.84
cTLHMM														
initiale	69.21	56.13	61.65	27.27	50.76	14.70	47.08	69.95	46.55	75.13	22.77	61.44	12.33	56.73
cyclique	69.21	51.33	67.42	23.72	58.36	14.50	47.73	69.95	44.93	77.84	19.28	72.56	12.32	56.78
perm.	69.21	51.29	67.47	22.92	60.39	14.55	47.57	69.95	43.84	79.78	18.60	75.22	12.32	56.78
GD-DG	69.21	51.20	67.59	21.62	64.02	14.53	47.63	69.95	43.47	80.46	17.59	79.53	12.31	56.82