ARCHITECTURES PARALLELES ET RESEAUX DE NEURONES ARTIFICIELS : MODELISATION PAR RESEAUX DE PETRI POUR UNE IMPLANTATION SUR UNE ARCHITECTURE MULTIPROCESSEUR A FLUX DE DONNEES

P. ABELLARD - S. CAMILLA

Laboratoire MS-LAIAT, Université de Toulon
BP132, 83957 La Garde Cedex

RESUME

Les réseaux de neurones artificiels présentent différentes propriétés qui les rendent particulièrement attractifs, comme par exemple, leur implantation sur des architectures parallèles qui permet de réduire considérablement les opérations très couteuses en temps nécessaires sur des machines informatiques classiques basées sur le modèle de Von Neuman. L'utilisation de modèles formels de représentation est nécessaire pour simplifier le développement des architectures parallèles. Les Réseaux de Petri à Flux de Données sont particulièrement bien adaptés à la modélisation des architectures à flux de données.

ABSTRACT

Artificial neural networks can achieve high computation rates by employing a massive number of simple processing elements. Neural networks with back-propagation connections provide a computing model capable of exploiting fine-grained parallelism to solve a rich class of optimization problems. In this paper, we propose a methodology applied to back-propagation algorithm and using Data Flow Petri Nets which are a powerful tool to modelize the parallelism. We describe the resolution method and explain the implementation on a multiprocessor machine using data flow processors NEC μPD 7281.

1. INTRODUCTION

Depuis plusieurs années, les réseaux de neurones artificiels font l'objet de nombreuses études car ils correspondent à une nouvelle approche pour la résolution d'un grand nombre de problèmes dont les solutions classiques ne sont pas toujours performantes (classification, traitement d'images, aide à la décision, optimisation). L'utilisation de ces algorithmes est une alternative prometteuse pour la résolution d'un grand nombre de problèmes, comme l'algorithme de la rétro-propagation du gradient. Aussi, à l'heure actuelle, la communauté scientifique et le monde industriel s'intéressent sérieusement au développement de ces techniques avec des machines multiprocesseurs.

2. ALGORITHME DE RETROPROPAGATION DU GRADIENT

Il s'applique aux réseaux de neurones de type multi-couches, suivant une technique d'apprentissage supervisée. C'est à dire qu'en phase d'apprentissage, on applique en entrée du réseau un ensemble de données auquel on fait correspondre une sortie désirée. Nous nous intéressons ici à l'algorithme qui modifie les poids à chaque présentation d'un couple entrée/sortie, et dont les principales étapes sont les suivantes :

- Initialisation des poids $W_{ij}^{[k]}$
- Présentation d'une donnée en entrée $v_i^{[k]}$
- Calcul de l'état du réseau par propagation

$$V_i^{[k]} = F\left[\sum_{j=1}^{n_{k-1}} W_{ij}^{[k]} V_j^{[k-1]}\right]$$

- Présentation du vecteur de sortie désirée (Sd) et calcul des gradients par rétro-propagation (F' est la dérivée de F)

$$\delta_i^{[k]} = 2\, F\left[\sum_{j=1}^{n_{k-1}} W_{ij}^{[k]} V_j^{[k-1]}\right](Sd_i - V_i^{[k]}) \quad \text{pour les cellules de sorties}$$

$$\delta_i^{[k-1]} = F'\left[\sum_{j=1}^{n_{k-1}} W_{ij}^{[k]} V_j^{[k-1]}\right]\sum_{m=1}^{n_k} W_{mi}^{[k]} \delta_m^{[k]} \quad \text{pour les autres cellules}$$

- Modification des poids $W_{ij}^{[k]} \leftarrow W_{ij}^{[k]} + \lambda \delta_i^{[k]} V_j^{[k-1]}$ avec λ le pas du gradient associé aux cellules, k l'indice de couche du réseau et n_k le nombre de neurones de la couche k.

3. LES ARCHITECTURES MULTIPROCESSEURS A FLUX DE DONNEES

L'architecture à flux de données (figure 1) ne comporte ni unité de commande, ni mémoire globale. Grâce au contrôle issu des flux de données et à l'exploitation de chaque opération comme fonction, ce type de machine permet à la fois de limiter le coût de synchronisation (gestion et contrôle réduits) et d'obtenir un haut degré de parallélisme.

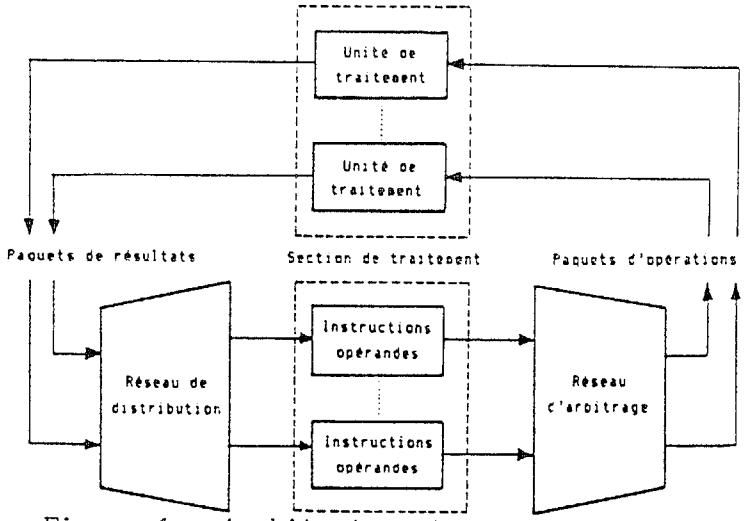


Figure 1 : Architecture à flux de données.

La spécification d'un système parallèle nécessite un langage et des formalismes purement fonctionnels permettant de décrire les relations temporelles (actions, communications,...) ou celles décrivant l'état du système. Parmi les formalismes existants, les Réseaux de Petri à Flux de Données sont particulièrement bien adaptés.

4. MODELISATION PAR RESEAUX DE PETRI A FLUX DE DONNEES (RdPFD)

La figure 2 montre que les RdPFD permettent de modéliser les réseaux de neurones. Dans un RdPFD, une opération est réalisée par un opérateur qui dispose d'un ensemble de données "Opérandes" et qui fournit un ensemble de résultats. Il contient deux sortes de places (variables et opérateurs) et se représente comme indiqué sur la figure 2

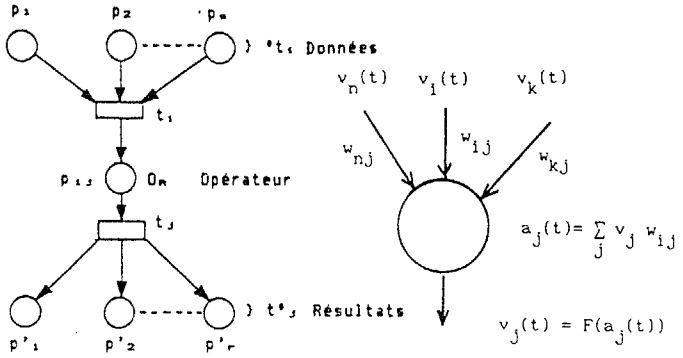


Figure 2 : Réseau de Petri à Flux de Données et réseau de neurones.

5. METHODOLOGIE.

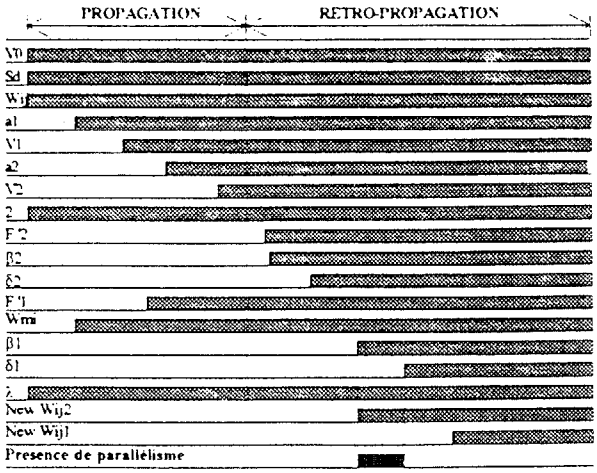
L'implantation d'algorithmes sur une architecture parallèle n'est pas un simple exercice de style. En effet, si sur une machine de type Von Neuman, la connaissance de l'algorithme suffit généralement pour sa mise en oeuvre, il n'en est pas de même sur une architecture parallèle. Tout d'abord, les algorithmes eux-mêmes ont été conçus à l'origine pour fonctionner sur des machines séquentielles. Cela nécessite alors une réflexion sur l'algorithme que l'on désire implanter pour voir s'il se prête bien à la parallélisation ou non. Il est important alors d'examiner en tout premier lieu le contexte applicatif de l'algorithme en question. Dans notre cas, il s'agit de réseaux de neurones pour lesquels nous savons que ce sont des processus distribués et fortement connectés qui se prêtent bien à une implantation sur des architectures dont le parallélisme peut être considéré à différents niveaux.

6. LE PARALLELISME DES RESEAUX DE NEURONES

On peut distinguer plusieurs niveaux de parallélisme dans les algorithmes neuronaux

- parallélisme des exemples d'apprentissage,
- parallélisme au niveau des couches,
- parallélisme sur les neurones et les connexions.

Le graphe des états ci-après est simple à construire. Au début de la phase de propagation, plusieurs données sont connues : les sorties des neurones de la couche 0, les sorties désirées sur l'exemple présenté, les poids de l'ensemble des neurones ainsi que la constante et le pas du gradient λ qui est considéré comme fixe. Ensiute, la propagation avant se fait normalement avec le calcul des états du potentiel de neurone, de sa sortie,... Il est alors évident que la mise à jour des poids de la couche 2 peut se faire pendant que le gradient sur la couche 1 est calculé.



Ces différentes remarques mettent en évidence un parallélisme par couche implantable facilement sur une architecture multiprocesseurs à flux de données.

7. ARCHITECTURE A FLUX DE DONNEES.

Elle a été réalisée avec des processeurs à flux de données NEC μPD 7281 constitués de 9 blocs principaux (figure 3).

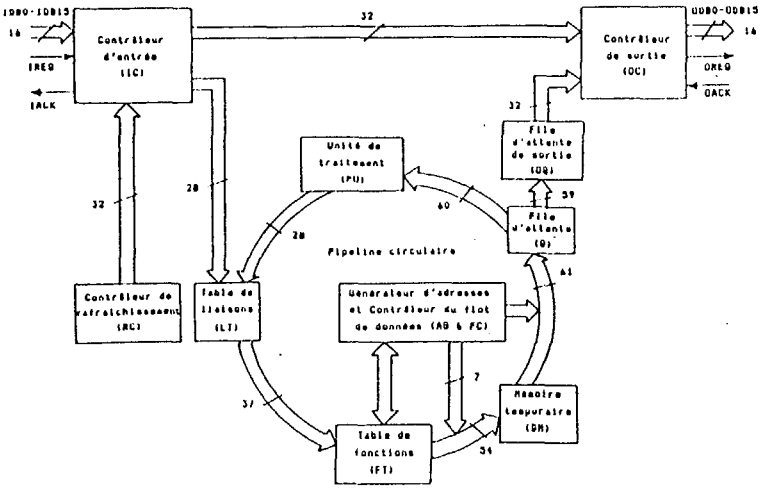


Figure 3 : Le processeur à flux de données

8. MODELISATION DE LA FONCTION DE TRANSFERT

Après calcul du potentiel V, le neurone actualise son état de sortie à travers une fonction d'activation. Le choix de cette fonction dépend de l'application à implanter ainsi que du mode d'apprentissage adopté. Dans les réseaux basés sur la rétro-propagation, on peut utiliser la fonction sigmoïde $Ft(V) = \frac{1 - e^{-\beta V}}{1 + e^{-\beta V}}$ avec $\beta/2$ la pente de la courbe à l'origine

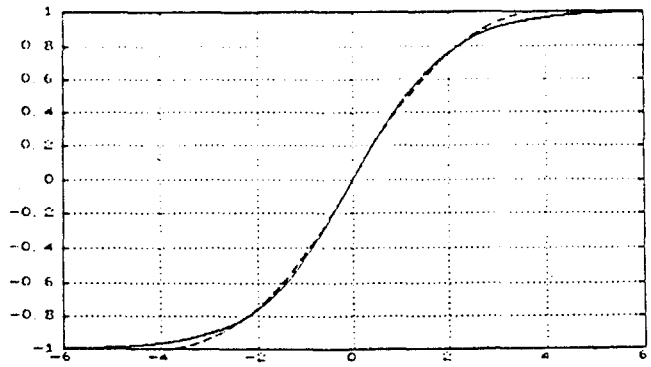


Figure 4 : Approximation de la sigmoïde.

Ce choix a pour avantage un calcul simple de la fonction dérivée. Lorsqu'une quantité de mémoire importante est disponible, la solution d'échantillonnnage présente plusieurs avantages :

- simplicité de représentation,
- rapidité (le temps d'un accès mémoire),
- aucune ressource spécifique.

9. MODELISATION PAR RESEAUX DE PETRI A FLUX DE DONNEES

Nous considérerons dans le cadre de cet article, uniquement l'exemple simple de la modélisation d'un réseau de neurone de bas niveau nécessaire à la résolution du XOR qui est le cas typique du problème non-linéairement séparable résolu par un réseau multicouche (figure 5). Ce réseau est composé de trois couches : une d'entrée avec deux cellules, une cachée avec deux cellules et une de sortie avec une seule cellule. Compte tenu de la parallélisation de l'algorithme, nous allons donc définir les opérateurs permettant la modélisation de la rétro-propagation à 2 niveaux (micro et macro-opérateurs) :

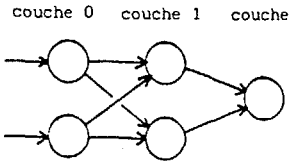
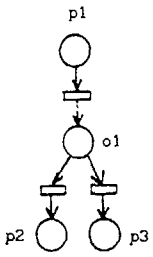


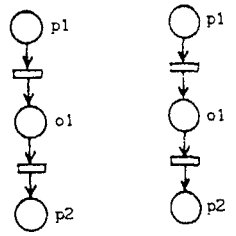
Figure 5 : Reseau pour le XOR

Figure 7 : Opérateur CONV(n)



Xp2 = F(Xp1)
Xp3 = F'(Xp1)

Figure 6 : Opérateur FONC.



$Xp2 = \sum_{n} Xp1$

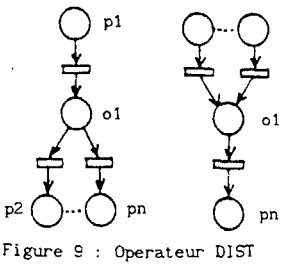


Figure 9 : Operateur DIST

Figure 8 : Operateur COP(n)

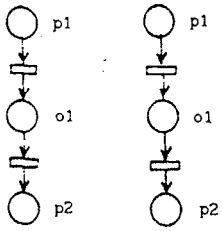


Figure 10 : Operateur MULPLX

Figure 11 : Operateur READ(Op,Q,t)

Figure 12 : Opérateur WRITE(Q,t)

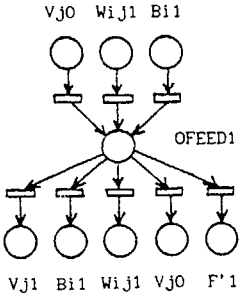


Figure 13 : Macro-opérateur OFEED1

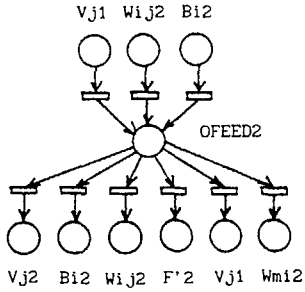


Figure 14 : Macro-opérateur OFEED2

- opérateur fonction FONC (figure 6) qui réalise le calcul de la fonction et de sa dérivée,

- opérateur convolution CONV(n) qui réalise la convolution de n variables successives (figure 7),

- opérateur copie COP(n) qui réalise n fois la copie de p1 dans p2 tout en attendant que p2 soit vide (figure 8),

- opérateur distribution DIST qui distribue cycliquement les jetons arrivant dans p1, dans l'ensemble des places de sortie : le premier dans p2, le deuxième dans p3....(figure 9),

- opérateur de multiplexage MULPLX qui réalise la copie de p1 dans pn, puis p2 dans pn....(figure 10),

- opérateur de lecture READ(Op,Q,t) qui effectue l'opération Op en retenant en premier membre la valeur dans p1 et comme second membre la valeur lue dans un tableau Q auto-incrémenté de dimension t. Le résultat est alors rangé dans p2 (figure 11),

-opérateur d'écriture WRITE(Q,t) qui effectue t fois, l'écriture en mémoire (tableau Q) de la valeur contenue dans p1 (figure 12).

Ces micro-opérateurs de base permettent la définition des macro-opérateurs utilisés pour la modélisation de l'algorithme :

- OFEED1 modélise la couche 1 en phase de propagation (figure 13),

- OFEED2 modélise la couche 2 en phase de propagation (figure 14),

- OBACK1 modélise la rétro-propagation pour la couche 1 (figure 15),

- OBACK2 modélise la rétro-propagation pour la couche 2 (figure 16).

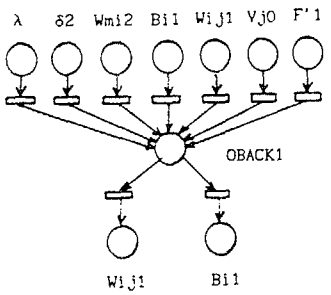


Figure 15 : Macro-opérateur OBACK1

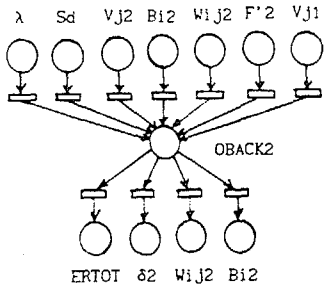


Figure 16 : Macro-opérateur OBACK2

La modélisation complète de l'algorithme de rétro-propagation du gradient peut alors s'effectuer suivant le Réseau de Petri à Flux de Données de la figure 17 directement implémentable sur l'architecture multiprocesseur utilisant les processeurs à flux de données NEC μPD 7281.

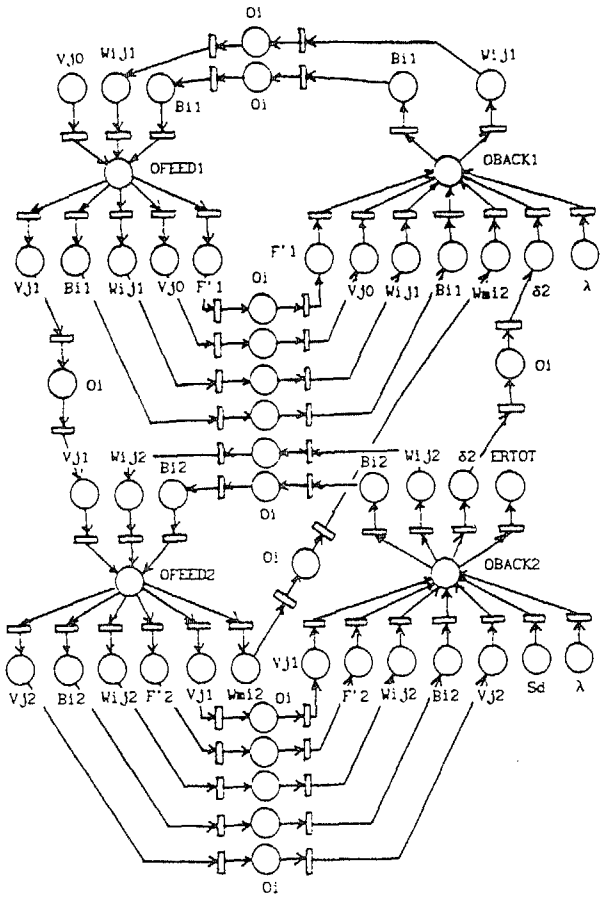


Figure 17 : Réseau complet

10. CONCLUSION

Si l'on veut effectuer, à moindre coût et avec efficacité des traitements parallèles sur une machine multiprocesseur, il est impératif de faire une étude approfondie du problème à résoudre. Pour cela, il est important d'avoir un modèle formalisable qui permette de représenter et d'étudier tous les aspects du parallélisme. Car, c'est de ce modèle que dépendra la pertinence de la réalisation de l'architecture. Les Réseaux de Petri à Flux de Données sont parfaitement bien adaptés à la résolution de ces problèmes et permettent de prendre en considération bien d'autres aspects qui n'ont pas pu être abordés dans ce papier comme la partition des exemples d'apprentissage, la définition du nombre optimal de neurones sur la couche cachée et l'auto-adaptation dynamique du grain.

MOTS CLES : Réseaux de neurones, réseaux de Petri, flux de données, architectures multiprocesseurs.

BIBLIOGRAPHIE

E. DAVALO, P. NAIM : *Des réseaux de neurones.* Editions Eyrolles 1990.

J. ALMHANA : *Modélisation par Réseaux de Petri à Flux de Données. Application à la synthèse de l'opérateur de Riccati rapide.* Thèse de Doctorat, Marseille, Juin 1983.