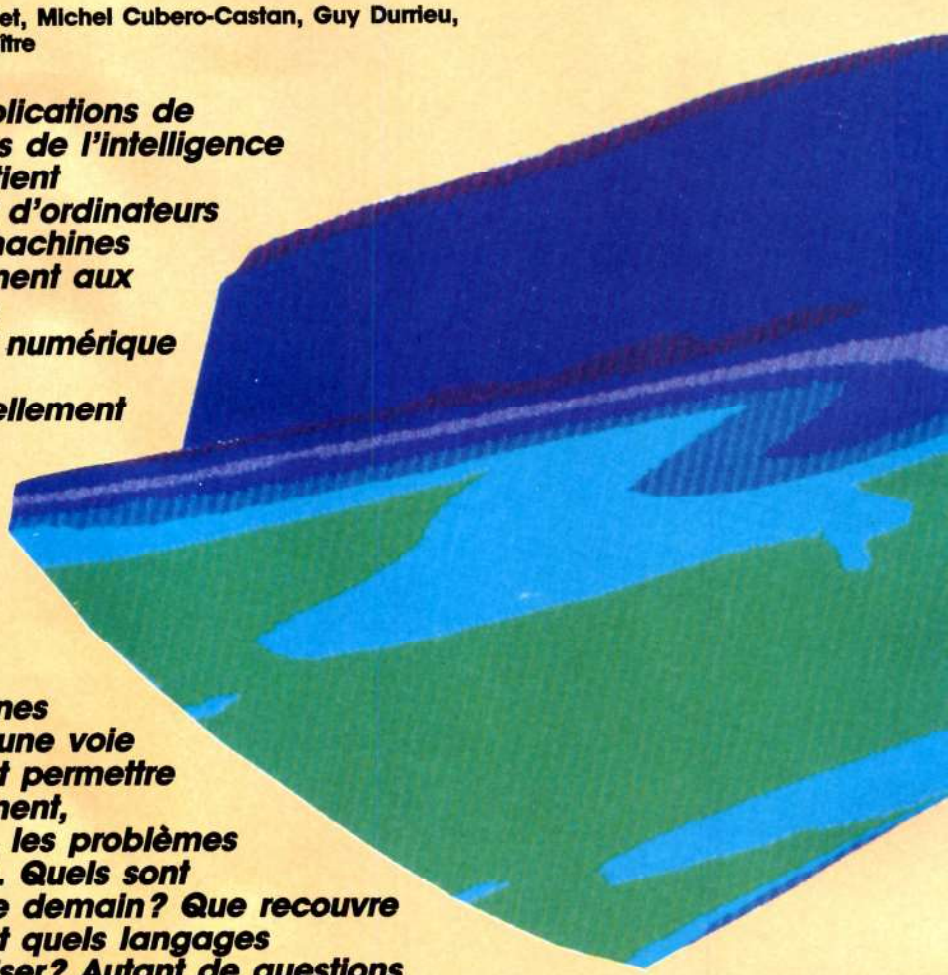
Les machines à réduction

par Eric Cousin, Christophe Coustet, Michel Cubero-Castan, Guy Durrieu, Bernard Lécussan et Michel Lemaître



Comment traiter les applications de plus en plus nombreuses de l'intelligence artificielle? La réponse tient dans le développement d'ordinateurs spécifiques, appelés machines symboliques; contrairement aux ordinateurs traditionnels fondés sur le traitement numérique de l'information, et travaillant donc essentiellement sur des nombres, elles manipulent non seulement des nombres, mais aussi et surtout des symboles et des liens entre ces symboles. Parmi ces machines symboliques, les machines à réduction constituent une voie d'avenir: elles devraient permettre de traiter plus efficacement, c'est-à-dire rapidement, les problèmes d'intelligence artificielle. Quels sont donc ces ordinateurs de demain? Que recouvre le terme de réduction et quels langages informatiques faut-il utiliser? Autant de questions auxquelles les auteurs répondent dans cet article.

Depuis une trentaine d'années, l'intelligence artificielle, dont l'objet est d'étudier et de reproduire à l'aide d'ordinateurs une partie des comportements humains, prend peu à peu sa place dans le paysage informatique. Aujourd'hui, elle revendique cette place à part entière : ses applications sont en effet de plus en plus nombreuses, et sortent des laboratoires. Ainsi l'industrie commence-t-elle à développer des systèmes qu'elle intègre par exemple dans ses procédés de fabrication. C'est le cas des systèmes experts, ces logiciels destinés à reproduire le raisonnement d'un spécialiste dans un domaine très pointu, et qui mettent à la disposition d'utilisateurs moins expérimentés une « expertise » que seules quelques personnes détiennent. Mais les applications de l'intelligence artificielle concernent aussi le traitement de la vision — c'est le problème de la reconnaissance des formes —, celui du langage naturel — comment une machine peut-elle comprendre une phrase écrite en langage commun ? —, ou encore celui de la reconnaissance vocale.

Quels ordinateurs doit-on construire pour supporter de telles applications ? La question est cruciale. Il s'avère en effet que face à la croissance des besoins en performances, notamment la rapidité d'exécution, et la sophistication des applications, les ordinateurs traditionnels, fondés sur le traitement numérique de l'information et manipulant donc essentiellement des nombres, vont se révéler de moins en moins adaptés.

On se trouve effectivement confronté à un type de traitement de l'information différent, sur lequel nous reviendrons en détail : le traitement symbolique. On manipule ici non seulement des nombres, mais aussi des symboles et des liens entre ces symboles, par exemple les mots d'un langage et leurs relations grammaticales.

De l'avis de tous les spécialistes, il devient donc urgent d'envisager des ordinateurs nouveaux, dédiés à ce type de traitement : ces « machines symboliques » permettraient alors de résoudre plus rapidement les problèmes cités plus haut. Du reste, une grande activité règne autour de telles machines dans les laboratoires de recherche ; de nombreux travaux ont lieu aux Etats-Unis, en Grande-Bretagne et dans une moindre mesure en France. Au sein de cette famille d'ordinateurs se trouvent les machines à réduction. Celles-ci, comme de nombreuses machines symboliques, s'appuient sur une programmation dite fonctionnelle, permettant d'exprimer la résolution d'un problème sous une forme particulière : la composition de fonctions, au sens mathématique du terme. Intérêt de ces machines : elles permettent de traiter encore plus efficacement les problèmes d'intelligence artificielle (fig. 1). Elles se pré-

et l'intelligence artificielle

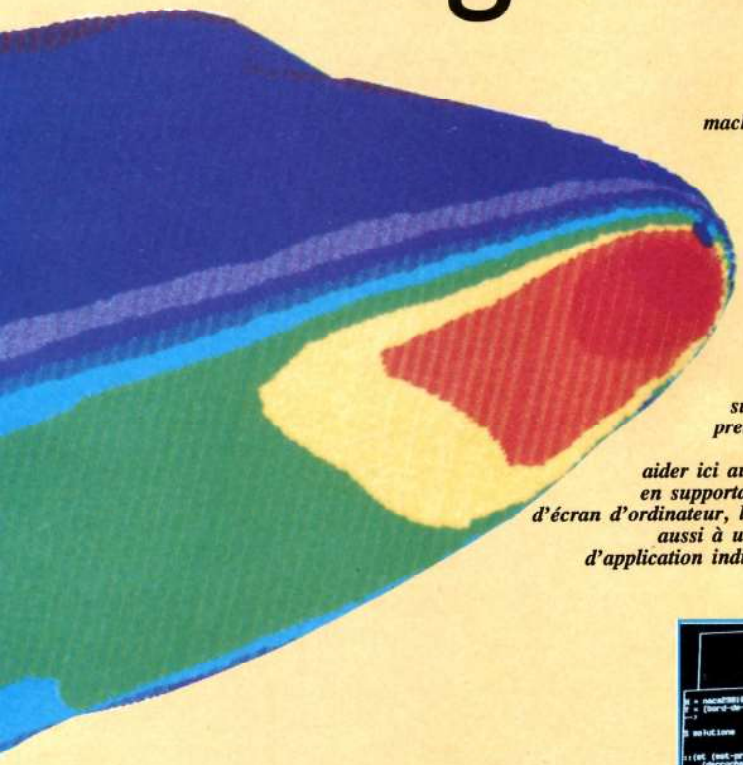
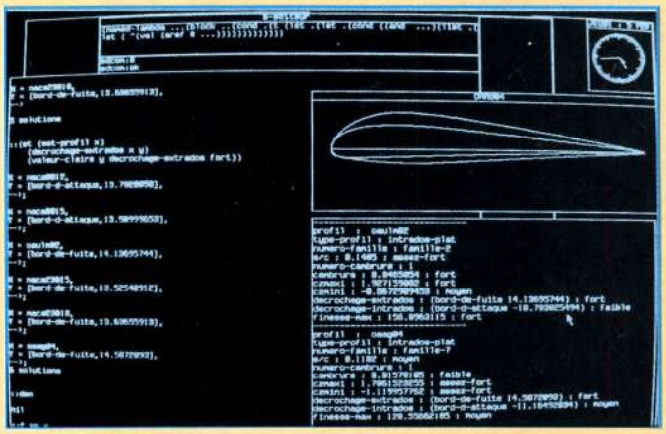


Figure 1. Les machines à réduction sont des machines dites symboliques. Il s'agit d'ordinateurs particulièrement adaptés au traitement symbolique de l'information, caractéristique de l'intelligence artificielle. Encore dans les laboratoires, elles devraient autoriser l'informaticien à traiter plus efficacement qu'avec les autres machines symboliques, c'est-à-dire plus rapidement, certaines applications de l'intelligence artificielle. Ainsi la machine MaRS, dont une maquette est en cours de réalisation au Centre d'études et de recherches de Toulouse (CERT), devrait permettre de faire « tourner » plus rapidement des systèmes experts — ces logiciels qui simulent le raisonnement d'un spécialiste —, par exemple pour l'aide à la conception de programmes de calculs aéronautiques. On voit sur la photo de gauche le résultat d'un calcul de répartition des pressions sur une navette du type Hermès faits à l'Office national d'études et de recherches aérospatiales; MaRS pourrait aider ici au choix des principes de calcul les mieux adaptés au problème, en supportant un système expert. La photo du bas représente sur une vue d'écran d'ordinateur, le résultat de la conception d'un profil d'aile d'avion, grâce ici aussi à un système expert, développé au CERT. Voilà donc un domaine d'application indiqué pour MaRS, et pour les machines à réduction en général. (Clichés AMD-BA et CERT)



Eric Cousin, ingénieur ENSEEIHT, est stagiaire de thèse au Centre d'études et de recherches de Toulouse (CERT).
Christophe Coustet, titulaire d'un diplôme d'études approfondies en informatique, est stagiaire de thèse au CERT.
Michel Cubero-Castan, docteur d'Etat en informatique (Université Paul Sabatier de Toulouse), est ingénieur de recherche au CERT.
Guy Durrieu, docteur de troisième cycle (Université Paul Sabatier), est ingénieur de recherche au CERT.
Bernard Lécussan, docteur d'Etat en informatique (Université Paul Sabatier), professeur à l'ENSAE, est chef du département informatique de l'ENSAE.
Michel Lemaître, ingénieur ENSEEIHT, docteur ingénieur en informatique, est ingénieur de recherche au CERT.

sentent ainsi comme l'une des grandes voies d'avenir dans ce domaine. Malgré des recherches peu nombreuses en France, un projet caractéristique de réalisation d'une machine à réduction est mené depuis 1984 au Centre d'études et de recherches de Toulouse (CERT): il s'agit de la machine MaRS (machine à réduction symbolique), dont une maquette doit voir le jour à l'été 1989.

Histoire de symboles.

On a coutume d'opposer le traitement symbolique et le traitement numérique — tout en sachant qu'une application informatique est généralement composée de ces deux types de traitement. Ce clivage, en quelque sorte, s'explique par l'histoire, en fonction des besoins du traitement de l'information. L'informatique fait en effet ses premiers pas au XIX^{e} siècle grâce au mathématicien anglais Charles Babbage, qui invente les concepts d'ordinateur et de programmation, dans la louable intention de libérer ses collègues du calcul fastidieux, et de surcroît fréquemment entâché d'erreurs, des tables astronomiques qui leur étaient nécessaires. Constatant le caractère uniforme et répétitif de cette activité, il introduit les premiers principes d'automatisation du calcul. Plus tard, les ordinateurs électroniques dignes de ce nom émergent, d'une part grâce aux progrès réalisés dans le domaine des machines à calculer électromagnétiques, qui n'utilisent donc pas encore les transistors, mais les relais électromagnétiques, et d'autre part du fait des besoins militaires en calculs numériques — tables balistiques, analyse d'informations cryptées, technique nucléaire naissante, etc., particulièrement durant la Seconde Guerre mondiale.

Il n'est pas étonnant, en conséquence, de trouver dans l'architecture même des calculateurs actuels, c'est-à-dire leur construction et la façon de les faire fonctionner, la marque de cette orientation première : l'accent est mis d'emblée sur la manipulation des nombres. Le concepteur s'est d'abord attaché à définir une représentation adéquate de ces nombres selon leur nature : format dit à virgule fixe pour les nombres entiers, ou à virgule flottante pour les nombres réels... Le mot clé est la précision de la représentation de ces nombres, surtout lorsqu'ils comportent des chiffres après la virgule dont l'arrondi peut entâcher d'erreur le résultat. L'effort a porté ensuite sur l'élaboration d'organes arithmétiques capables de traiter ces nombres de façon efficace, c'est-à-dire rapide. On a enfin observé que le calcul numérique faisait souvent intervenir, par exemple pour des calculs matriciels, une structuration régulière des données mises en mémoire, de taille prévisible, mettant en jeu des élé-

ments juxtaposés désignés par un indice et traités de manière identique. C'est ainsi qu'est né le modèle architectural dit de von Neumann (voir « L'architecture des nouveaux ordinateurs » dans ce numéro), avec son processeur de traitement des données et sa mémoire linéaire, composée d'un certain nombre d'unités élémentaires, cellules identiques et contiguës pouvant contenir instructions et données.

Qu'en est-il des symboles dans tout cela ? En fait la notion de symbole existe déjà, mais sous la forme rudimentaire de nom de variable, c'est-à-dire d'entité dont la seule finalité est de posséder un contenu qui est utile au déroulement du programme, le nom de la variable, lui, n'étant pas utile. Ce nom permet d'identifier un tableau, le résultat d'un calcul, etc. L'existence des symboles est ici purement liée au « confort » du programmeur : il est en effet plus clair pour ce dernier de récupérer le solde d'un compte bancaire à l'aide d'une variable portant le nom SOLDE plutôt que de faire référence à une cellule mémoire par son numéro.

On le voit, de tels symboles ne sont pas nécessaires au traitement lui-même. On devra attendre la fin des années 1950 pour voir émerger des langages de programmation adaptés au traitement symbolique. John Mc Carthy, chercheur au Massachusets Institute of Technology crée alors le langage Lisp[1], devenu depuis lors l'un des outils fondamentaux d'expression des problèmes de l'intelligence artificielle. Plus tard, en 1971, le Français Alain Colmerauer, de l'université de Marseilles, devait, avec Prolog, donner naissance à une autre famille de langages symboliques.

Pour Lisp, on parle de programmation fonctionnelle ; c'est sur ce type de langage que s'appuient les machines à réduction. En programmation fonctionnelle, on dit aussi applicative par référence à l'application mathématique, un programme se présente sous la forme de fonctions, au sens mathématique du terme, définies par composition de fonctions élémentaires connues de la machine et/ou des fonctions que l'on définit soi-même. Comme en mathématiques, on définit alors la fonction f qui fait correspondre à l'argument x le résultat y, en écrivant : f(x) = y. Par exemple, on peut considérer la fonction « mettre au pluriel » et l'appliquer à n'importe quelle phrase. La phrase au singulier est symbolisée par l'argument x et la phrase au pluriel par le résultat y.

Pour Prolog, on parle de programmation déclarative. On exécute alors un programme en donnant une série d'énoncés ou de propositions au sens logique du terme, par exemple « le père du père de quelqu'un est son grand-père ». En attribuant à chacun d'entre eux une valeur logique « vrai » ou « faux » en fonction d'un contexte particulier, on aboutit au résultat du programme, dont par exemple le but peut être de savoir : « Quels sont tous les grands-pères de Bernard ? »

Quelles machines ?

Que la programmation soit fonctionnelle ou déclarative, le traitement symbolique se distingue du traitement numérique par une plus grande complexité des données, entraînant par là-même une plus grande complexité de leurs traitements. On conçoit que pour un programme analysant des textes en langue naturelle pour prendre cet exemple, les

DRÔLES DE MÉMOIRES

Le traitement symbolique se distingue du traitement numérique, notamment par une forte complexité de la représentation des données dans la structure interne de l'ordinateur. Considérons la représentation dans une machine numérique d'une instruction écrite en langage de programmation Pascal, qui s'écrit : « tab : array [1:1000] of integers ». Cela signifie : « définir un tableau de mille nombres entiers ». Ce tableau se traduit une fois pour toutes dans du silicium par une juxtaposition contiguë de cellules de mémoire qui contiennent chacune un entier. Les adresses, numérotées de 2 001 à 3 000 en A, sont en quelque sorte les numéros des cellules auxquelles il faut se rendre pour accéder à l'information, à savoir tel ou tel entier. Pour se rendre à une adresse précise, comme le montre la flèche verticale, l'ordinateur est obligé de « balayer » toutes les adresses qui la précèdent. Les parties hachurées sont des cellules vides. Cette juxtaposition figée, découlant de la régularité de la structure de données, est simple à réaliser ; le traitement de ces données est lui-même simple.

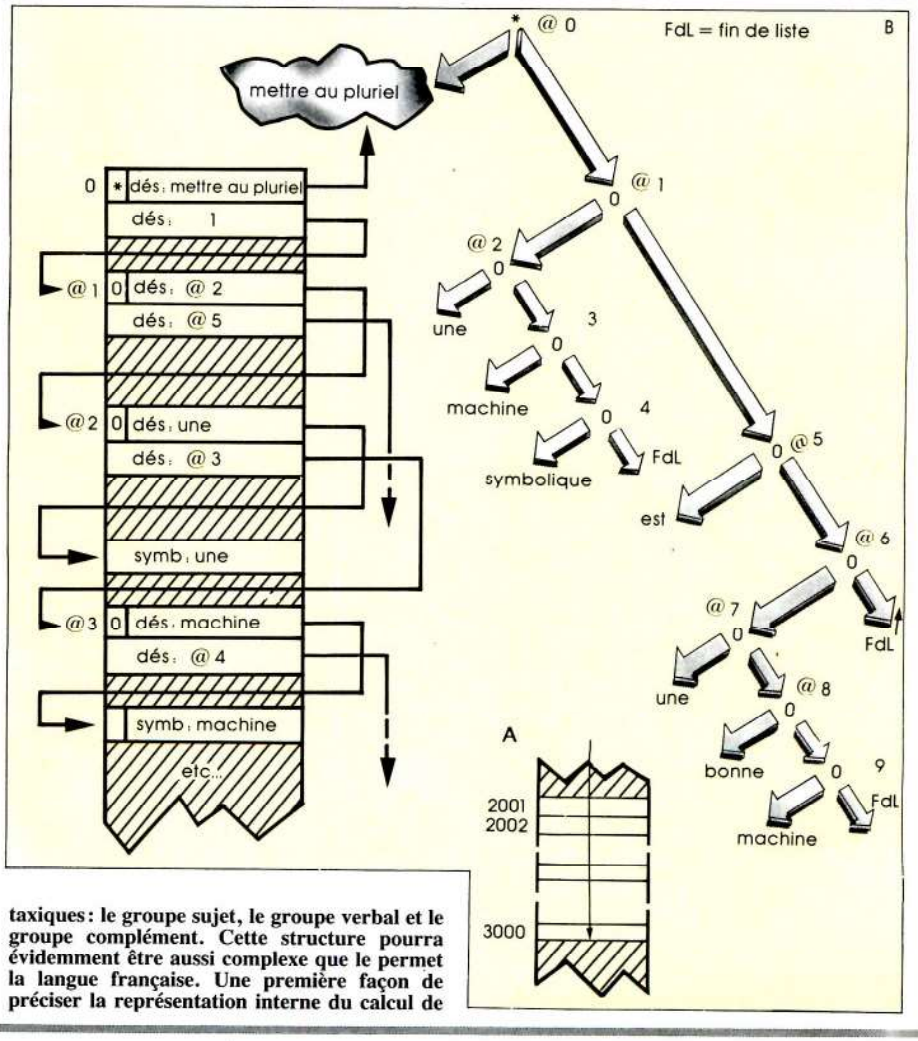
En revanche, dans une machine de traitement symbolique, on doit faire face à une forte irrégularité des structures de données. Nous voyons sur le schéma B une façon de représenter en mémoire, précisément dans le cadre d'une machine fonctionnelle, une phrase et l'application d'une fonction à cette phrase. Considérons ainsi le « calcul » par l'ordinateur de la fonction suivante (ce qui revient à exécuter un programme) :

mettre au pluriel (« une machine symbolique est une bonne machine »).

Une syntaxe plus proche de celle utilisée dans beaucoup de langages fonctionnels pourrait être :

(mettre au pluriel '((une machine symbolique) est (une bonne machine)))

Sous cette forme apparaît en premier lieu la fonction à appliquer puis, encadré par '( ), l'argument sur lequel elle doit s'appliquer. La mise entre parenthèses apparaissant à l'intérieur de la phrase argument explicite la structure de cette dernière en trois éléments syntaxiques : le groupe sujet, le groupe verbal et le groupe complément. Cette structure pourra évidemment être aussi complexe que le permet la langue française. Une première façon de préciser la représentation interne du calcul de



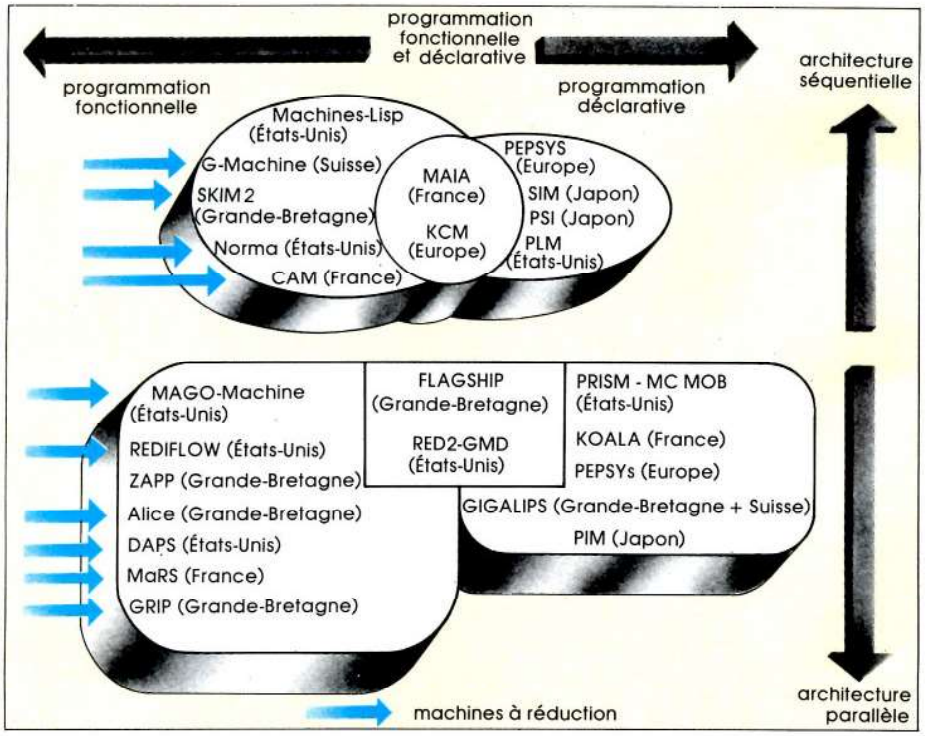


Figure 2. Depuis une quinzaine d'années, les projets de recherche ayant ou devant donner lieu à la réalisation de maquettes ou de machines de traitement symbolique, donc dédiées à l'intelligence artificielle, sont nombreux: ils sont, comme on le voit sur cette figure, américains, britanniques, et dans une moindre mesure français, japonais ou parfois le fruit d'une collaboration à l'échelle européenne.
Nous présentons ici ces machines d'une part en fonction du type de programmation sur lequel elles sont fondées: certaines utilisent la programmation fonctionnelle avec des langages de la famille Lisp, d'autres la programmation déclarative, caractéristique de l'autre famille de langages de l'intelligence artificielle, la famille Prolog; quelques-unes utilisent les deux à la fois (partie centrale de la figure).
Nous les présentons aussi en fonction de leur architecture: il y a les machines séquentielles, ne permettant de traiter les informations que les unes après les autres, et n'utilisant qu'un seul processeur, et les machines parallèles, ou multiprocesseurs, capables d'effectuer certaines tâches en simultané.
Nous voyons que les machines à réduction (en couleur), cette catégorie de machines de traitement symbolique destinées à effectuer le plus efficacement, c'est-à-dire rapidement, un programme d'intelligence artificielle, sont des machines fonctionnelles et proviennent des Etats-Unis, de Grande-Bretagne ou de France. (Schéma d'après auteurs)

cette fonction consiste à le figurer sous forme d'arborescence (la partie droite du schéma B). On y distingue un certain nombre de « nœuds » représentés par « * » ou « 0 », et identifiés par ce que l'on appelle des étiquettes de la forme « @ », qu'on pourra ensuite assimiler à des adresses placées elles-mêmes dans des cellules de mémoire. Outre ces nœuds, on rencontre ensuite des « feuilles » qui sont les symboles présents dans le calcul de la fonction. Un nœud est introduit à chaque parenthèse « ouvrante » ou à chaque nouveau symbole rencontré. La machine exécute donc le programme en suivant en quelque sorte les flèches constituant l'arbre. La partie gauche du schéma B précise la façon dont cet arbre est implanté dans la structure interne de la machine, la mémoire.

Un nœud est composé de deux cellules adjacentes, et une feuille d'une cellule seulement. Les cellules composant le nœud constituent ce que l'on appelle des « désignateurs » (dés:), c'est-à-dire contiennent une information permettant de localiser ce qui suit le nœud dans l'arborescence; ils correspondent aux flèches dans la partie gauche du schéma B, qui elles-mêmes suivent la démarche arborescente du programme. On trouve également dans un nœud une information permettant de déterminer s'il s'agit d'un nœud de type « * » ou « 0 ». Les cellules correspondant aux feuilles sont identifiées par (symb.).

Les zones hachurées indiquent que les différents composants de l'arbre ne sont en général pas placés dans des cellules de mémoire contiguës, contrairement au cas précédent en A. En effet lorsqu'un nouveau nœud doit être créé, quand par exemple la fonction aura été appliquée et le résultat exploité, la machine cherche alors le premier emplacement disponible. Et inversement, lorsqu'une cellule n'est plus nécessaire, elle redevient libre. Entre les différents éléments de l'arbre peuvent donc à un instant donné se trouver des cellules libres. On le voit, la structure de données et leur traitement sont beaucoup plus complexes que dans le cas d'une machine de traitement numérique.

traitements nécessaires diffèrent fortement en raison de la diversité des mots que l'on peut employer et des dispositions possibles pour chacun d'eux au sein de la phrase. Par suite, le comportement d'un programme symbolique est difficilement prévisible, notamment quant à ses besoins en ressources: temps de calcul, espace mémoire dans l'ordinateur. Par ailleurs, les structures de données se caractérisent, contrairement au traitement numérique, par une forte irrégularité dans leur organisation au sein d'une mémoire.

Ainsi, comme le montre dans l'encadré le schéma A, un tableau de mille nombres entiers sera représenté physiquement une fois pour toutes dans la structure interne d'une machine de traitement numérique, par une juxtaposition régulière de mille cellules de mémoire contiguës, numérotées ici de 2 001 à 3 000, sur un morceau de silicium. Toutes les cellules contiennent l'information relative à un nombre entier du tableau, en suivant l'ordre des nombres. Somme toute, une représentation simple à réaliser.

En revanche, dans une machine de traitement symbolique, comme le montre le schéma B, la disposition des cellules de mémoire sur le silicium n'est pas régulière, et peut présenter des espaces vides, car le « calcul » de certaines expressions dans un programme peut nécessiter d'allouer des cellules de mémoire, qui ne seront plus utiles dès que cette expression aura été calculée et son résultat exploité par le programme. Les cellules concernées sont alors réutilisables.

Pour être efficace, donc rapide, une machine doit être adaptée à ce type de traitement: les ressources en mémoire doivent être gérées dynamiquement, c'est-à-dire pendant le calcul, et non statiquement avant le traitement. Et de fait, les projets ayant donné lieu ou devant donner lieu à des réalisations de maquettes ou de machines symboliques depuis une quinzaine d'années sont nombreux (fig. 2). Qu'ils utilisent, d'ailleurs, la programmation fonctionnelle ou déclarative; qu'ils soient bâtis sur une architecture mono-processeur, c'est-à-dire ne permettant de traiter les informations que séquentiellement, les unes après les autres, ou multi-processeur, c'est-à-dire effectuant certaines tâches en simultané. Mais dans ce flot de machines, celles dites à réduction présentent un intérêt particulier: l'utilisation conjointe de langages fonctionnels et d'un modèle adapté à l'exécution des programmes écrits dans ces langages, la réduction, que nous allons détailler plus loin, permettent d'exécuter avec encore plus d'efficacité, c'est-à-dire de rapidité, un programme d'intelligence artificielle.

La réduction à l'honneur.

L'idée est la suivante. D'un côté, la programmation fonctionnelle, qui est fondée sur une théorie mathématique: le lambda-calcul. De l'autre, une autre théorie mathématique, la logique combinatoire, datant des années 1930, et qui savère facile à mettre en œuvre au niveau matériel. Or, en 1958, deux logiciens américains montraient l'équivalence entre ces deux théories! Lambda-calcul et logique combinatoire se rejoignant, il est dès lors aisé de mettre en œuvre les mécanismes de la logique combinatoire pour exécuter un langage fonctionnel. Ce sont ces mécanismes qui sont ceux de la réduction symbolique. Entrons dans le détail. Les langages applicatifs tels Lisp sont issus d'une théorie mathématique, le lamb-

(1) J. Mc Carthy, « Recursive functions of symbolics expressions and their computation by machine », *C.ACM, 3*, n° 4, 184, 1960.

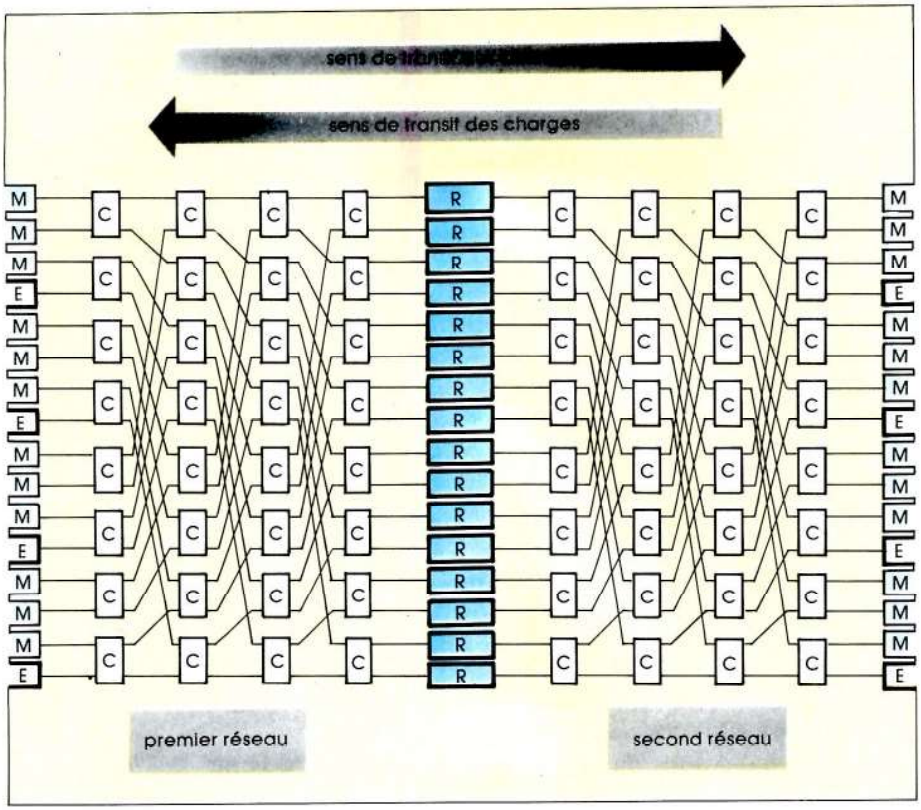


Figure 3. Sur cette figure est schématisée l'architecture, c'est-à-dire la façon dont sera construite et dont fonctionnera la machine MaRS. Cette dernière sera une machine parallèle : elle sera en effet composée de seize processeurs de réduction indépendants (R). Eléments centraux de la machine car à l'origine du gain en performance dans l'exécution d'un programme d'intelligence artificielle, ils permettent, en appliquant une fonction donnée (par exemple « mettre au pluriel ») sur un argument donné (dans le même exemple, une phrase), de transformer progressivement un programme, la forme obtenue à la fin de ce processus étant le résultat du programme original. La machine sera aussi constituée d'autres processeurs spécialisés dans la mémorisation des données (M), la communication entre processeurs (C) et l'échange des données avec l'extérieur (E) ou encore d'entrées-sorties. La figure indique d'une part le sens dans lequel transitent les données que la machine doit traiter (de la gauche vers la droite). Ainsi dans le premier réseau, celui situé à gauche des processeurs de réduction, transitent des données provenant de mémoires ou de processeurs d'échange avec l'extérieur. Les résultats ou les étapes de calcul sont ensuite acheminés à travers le second réseau. La figure indique aussi le sens de « transit des charges » (de la droite vers la gauche) : il s'agit, dans le réseau de gauche, d'informations qui permettent de savoir quels sont les processeurs disponibles pour effectuer la réduction et, dans le réseau de droite, d'informations qui indiquent aux processeurs de réduction quel est l'espace disponible dans chaque mémoire. Le processeur de communication est déjà réalisé. (Schéma d'après CERT)

da-calcul, développée en 1932 par l'Américain Alonzo Church dans le but de formaliser la notion de calcul, c'est-à-dire de décrire ce qui se passe quand, lors d'un calcul (la résolution d'un système différentiel par exemple) et par suite sa programmation, on remplace le nom d'une fonction par sa définition, et de décrire plus particulièrement ce que recouvrent la *substitution* et la *réduction*. La substitution consiste à remplacer les arguments formels (dans l'exemple cité plus haut, x) par les arguments réels (dans le même exemple, la phrase au singulier). La réduction consiste ensuite, pour exécuter le programme, à remplacer dans une expression donnée une opération par son résultat, c'est-à-dire appliquer une fonction sur un argument.

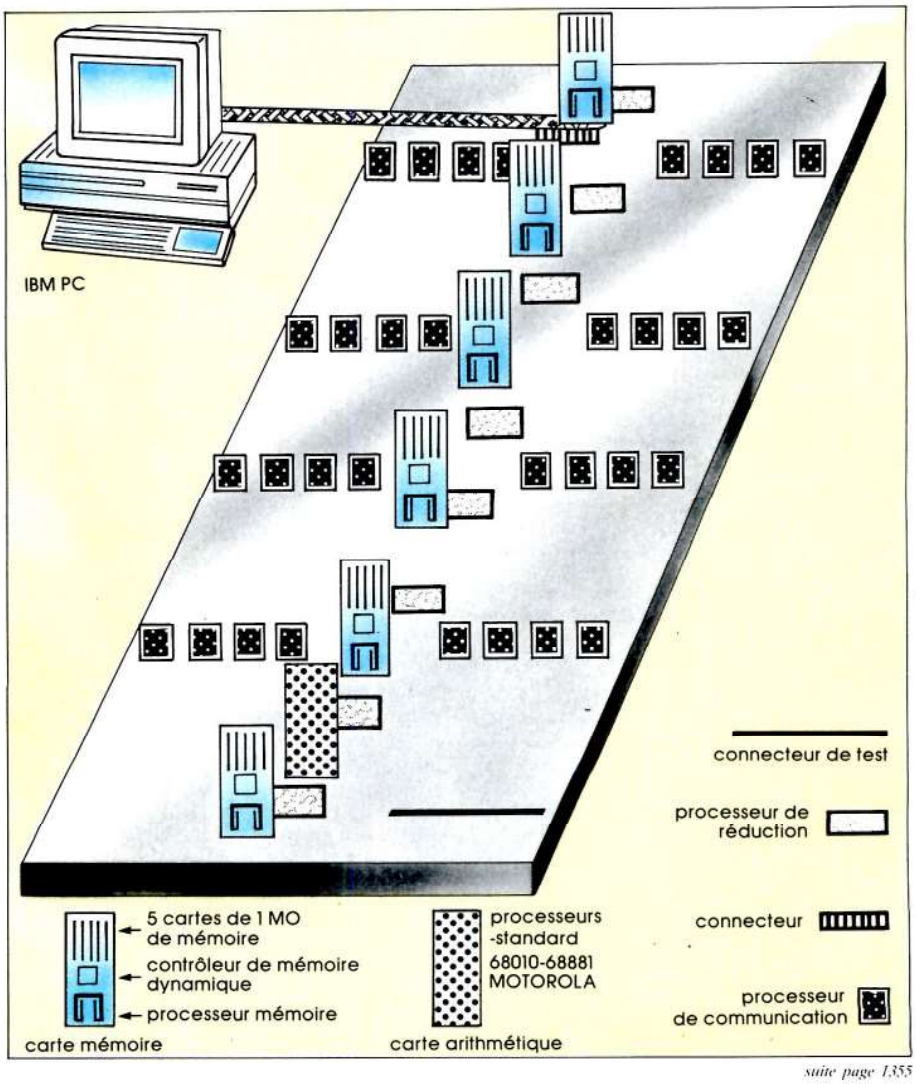
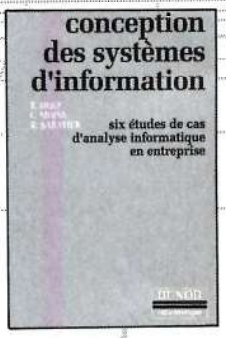


Figure 4. Le prototype de machine MaRS qui devrait voir le jour courant 1989 comprendra deux cartes du type de celle que nous présentons sur cette figure, et qui regrouperont l'ensemble des processeurs spécialisés dans les tâches de réduction, communication, mémorisation et échange avec l'extérieur. Chaque carte, comme celle-ci, portera huit processeurs de réduction, base du traitement symbolique. Mais aussi, en plus des processeurs d'entrées-sorties des données (non visibles ici), qui seront reliés par l'intermédiaire d'un connecteur et d'un câble à un micro-ordinateur IBM PC — servant en quelque sorte à piloter la machine —, une carte se composera de : six cartes mémoire (comprenant chacune le processeur de mémorisation lui-même, assorti de cinq cartes d'un méga-octet de mémoire et d'un « contrôleur dynamique » servant à gérer ces cinq cartes ; une carte de calcul arithmétique, nécessaire à la résolution des calculs numériques présents même en traitement symbolique ; trente-deux processeurs de communication, et un « connecteur de test », utilisé pour effectuer des tests sur le fonctionnement de la carte. (Schéma d'après CERT)

suite page 1355

LA TECHNOLOGIE DU FUTUR



TURBO PASCAL et son environnement
J. RIVIERE
Traitant de Turbo Pascal sous MS DOS, ce guide centré sur les versions 3 et abordant la nouvelle version 4.0, met en évidence les relations langage/système-machine. Illustré de nombreux exemples.
Dunod Informatique - 304p. - 160F

CONCEPTION DES SYSTÈMES D'INFORMATION
B. HERZ, C. MOINE, R. SABATIER
A partir de six cas d'entreprises, présentation des modèles et des moyens de conception des systèmes d'information avec une analyse de leur mise en œuvre permettant de traiter efficacement les problèmes rencontrés dans l'informatisation des entreprises.
Dunod Informatique - 320p. - 120F

CONSTRUIRE LES ALGORITHMES Les améliorer, les connaître, les évaluer
C. PAIR, R. MOHR, R. SCHOTT
Outil de formation et de perfectionnement proposant, à tous ceux qui maîtrisent déjà les règles élémentaires de la programmation, des moyens pour évaluer les algorithmes classiques et les méthodes pour les rendre plus performants.
Dunod Informatique - 248p. - 130F

ALGORITHMIQUE. 2. Structures de données et algorithmiques de recherche
P. BERLIOUX, P. BIZARD
Les méthodes, présentées dans le précédent ouvrage, trouvent ici leur application dans la résolution de problèmes classiques de structures de données (exemples en Pascal) et de parcours de graphes dont les algorithmes sont très utiles en Intelligence Artificielle et en recherche opérationnelle.
Dunod Informatique - 232p. - 130F

LOGICIELS INTERACTIFS ET ERGONOMIE Modèles et méthodes de conception
M.F. BARTHET
Indispensable à tous ceux qui s'intéressent à la qualité de l'interface homme-machine: conception de logiciels interactifs permettant d'intégrer les recommandations d'ergonomie et de faire évoluer le logiciel en fonction de tests sur prototypes ou en site réel.
Dunod Informatique/Cépia - 232p. - 170F

LE LANGAGE C
D. GALLAND
Manuel d'apprentissage et de référence du langage C, illustré de nombreux exemples concrets et de conseils de mise en œuvre: description du langage, utilisation des pointeurs et de la bibliothèque, développement du logiciel sous MS DOS...
Dunod Informatique - 248p. - 150F

LA STRUCTURE DES BASES DE DONNÉES
M. LEONARD
Construit autour du modèle relationnel de données et privilégiant les exemples, cet ouvrage pratique propose différents moyens de résolution permettant aux concepteurs et aux utilisateurs d'améliorer leur compréhension des bases de données et de les faire évoluer.
Dunod Informatique - 280p. - 170F

LES API. Architecture et application des automates programmables industriels
G. MICHEL
Présentation de l'API en tant que machine, système et outil, dans le contexte des logiques programmables et par rapport à ses domaines d'application.
356p. - 385F

SYSTEMES ASSERVIS NON LINÉAIRES
J. Ch. GILLE, P. DECAULNE, M. PELLEGRIN
Exposé didactique des plus importantes méthodes non linéaires, évaluant leur intérêt respectif et indiquant les conditions d'application.
228p. - 280F

MACHINES TOURNANTES ET CIRCUITS PULSÉS. Applications industrielles et médicales de l'analyse spectrale
L. BOURGAIN, R. DART, J. BOURGAIN
Synthèse novatrice proposant, à tous les techniciens confrontés aux problèmes de mécanique vibratoire, une connaissance conjointe de la théorie des vibrations et de la technique des mesures.
560p. - 240F

MÉTHODE DES PLANS D'EXPÉRIENCES
J. GOUPY
Une méthode qui va révolutionner l'expérimentation de toutes les sciences appliquées en obtenant des résultats plus fiables. Les outils mathématiques et statistiques de cette méthode sont mis en œuvre dans de nombreux exemples. Une application est entièrement développée sur lotus 1-2-3.
280p. - 180F

Dunod
POUR ALLER PLUS LOIN

suite de la page 1352

Pour réaliser l'opération de substitution décrite plus haut, les machines symboliques actuelles utilisent le principe dit d'environnement: on mémorise, dans une structure de données spéciale, la valeur associée à un instant donné à chaque argument. L'existence d'un tel environnement s'avère pénalisante du point de vue des performances. C'est pourquoi, en 1979, l'informaticien anglais David Turner propose un modèle d'exécution des programmes qui permet de s'affranchir du principe d'environnement en utilisant des résultats mathématiques de 1958[2]. Cette année-là, les deux logiciens américains Curry et Feys avaient montré l'équivalence entre le lambda-calcul et une autre théorie mathématique de formalisation du calcul, la logique combinatoire[3]. En logique combinatoire, il n'existe pas de notion d'argument formel. Le mécanisme de substitution précédent est partie intégrante de la fonction. Il est réalisé par des fonctions particulières constitutives de la précédente, appelées combinateurs (d'où le nom de logique combinatoire), dont l'exécution sert en fait à constituer progressivement l'expression à calculer. Comme en lambda-calcul, l'exécution d'un programme se fait par réductions successives, telles que nous les avons décrites plus haut. Les règles de réduction, élémentaires et peu nombreuses, définissent pas à pas la transformation progressive du programme avec ses données. A la fin de l'exécution, donc lorsque ce processus de transformation s'arrête, la forme obtenue est le résultat du programme original.

Le fait de pouvoir transformer tout langage applicatif en expressions combinatoires prend alors tout son intérêt quand Turner se rend compte, en 1979, qu'un ordinateur dont le langage machine, c'est-à-dire celui qu'il comprend directement, est constitué de combinateurs, peut être assez aisément réalisé. Il n'y a alors plus besoin du mécanisme d'environnement décrit plus haut. On obtient ainsi des gains de performance sur la vitesse, d'autant plus qu'en choisissant son propre jeu de combinateurs, on peut diminuer le nombre de réductions nécessaires à l'obtention du résultat du programme[4]. Le projet MaRS en est un exemple.

Toujours plus.

En outre, la logique combinatoire étant assimilable à un langage applicatif, ou fonctionnel, elle bénéficie de toutes les bonnes propriétés de ce type de langage. En particulier, le calcul d'une expression donne toujours le même résultat quelle que soit la façon d'effectuer les réductions. Il est notamment possible d'en effectuer plusieurs simultanément. Cela est très important: une machine à réduction possède donc intrinsèquement une grande possibilité de parallélisme. Une analyse automatique des programmes, que nous ne décrirons pas ici, permet en effet de savoir quels sont les arguments des fonctions qui sont absolument nécessaires au calcul. Le parallélisme s'obtient alors en lançant simultanément la réduction sur tous ces arguments, dits stricts. Si on représente de façon interne une expression constituée sous une forme particulière appelée graphe, permettant notamment de ne pas effectuer plusieurs fois les mêmes réductions, on augmente encore les performances. De plus, la synchronisation de l'ensemble peut être réalisée très simplement. On dit alors qu'on fait de la réduction de graphe. On le voit clairement, un tel modèle se prête à l'exploitation d'un traitement parallèle.

Or justement, malgré les progrès constants de la technologie, en vitesse d'exécution et en miniaturisation, soit en nombre de transistors intégrés sur une surface donnée de silicium, un gain important de performance, d'un facteur cent ou mille par rapport aux machines actuelles, qu'elles soient numériques ou symboliques, ne peut être envisagé qu'en exploitant le parallélisme de traitement qui existe dans une application. C'est d'ailleurs pourquoi beaucoup de travaux de recherche concernent actuellement la définition d'architectures de machines multiprocesseurs composées de plusieurs unités de traitement spécialisées coopérant pour résoudre un même problème.

L'introduction du parallélisme dans le traitement ne peut cependant se faire sans répondre à quatre questions fondamentales: comment découper le problème en sous-problèmes pouvant être traités séparément? Comment répartir ces sous-problèmes sur les différentes unités d'exécution? Comment synchroniser l'ensemble? Et enfin qui, du programmeur ou de la machine devra gérer le parallélisme? Le projet MaRS répond à ces quatre questions.

Un petit tour sur la planète MaRS.

Le projet de construction d'une machine à réduction au CERT a débuté en 1984. C'est le résultat d'une convergence de vues et d'efforts entre des chercheurs sur les langages et sur les architectures. Le terrain avait été préparé depuis longtemps par deux types d'études relativement éloignés l'un de l'autre, menés dans notre département d'informatique. D'une part une série d'études en génie logiciel nous avait persuadés de l'intérêt des langages applicatifs; d'autre part, un projet d'architecture de machine parallèle fondée sur un autre modèle d'exécution appelé « data-flow » (voir « De l'ordinateur classique au supercalculateur » dans notre numéro d'avril 1980) avait abouti à la construction d'une maquette. D'autres compétences toulousaines en architecture (sur des machines appelées M3L ou LTR) furent réunies, et l'étude d'architectures proprement dites commençait en 1985.

MaRS est fondée sur une architecture modulaire multiprocesseurs (plusieurs processeurs indépendants) et une réduction parallèle des programmes traduits en graphes de combinateurs, telle que nous l'avons décrite à l'instant. On répond ainsi à la première et à la troisième question posées plus haut. La gestion de l'en-

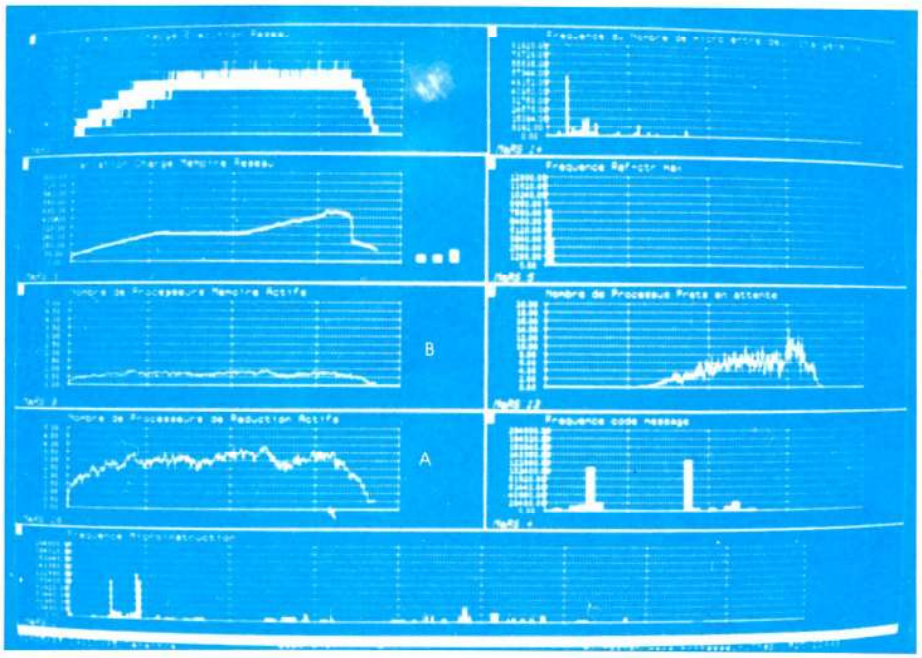


Figure 5. Cette vue d'écran est en quelque sorte le tableau de bord du simulateur de la future machine MaRS, réalisé au CERTet reproduisant le fonctionnement de MaRS tel qu'on le connaîtra lorsque la maquette sera construite. On y voit notamment le nombre de processeurs de réduction actifs tout au long de l'exécution d'un programme (A), ce qui donne le degré de parallélisme atteint, ainsi que le nombre de processeurs de mémorisation actifs (B). Cela permet d'estimer la performance atteinte par la machine. (Cliché CERT)

(2) D. Turner, « A new implementation technic for applicative languages », *Software practice and experience*, 9, 31, 1979.
(3) H.B. Curry, R. Feys, et W. Craig, *Combinatory Logic*, vol. 1, North Holland, 1958.
(4) Par les auteurs, « Towards the design of a parallel graph reduction machine: the MaRS project », *Graph reduction workshop*, septembre 1986, Santa Fe.

semble ne fait intervenir aucun mécanisme centralisé, de façon à prévenir tout goulet d'étranglement dans les échanges entre les processeurs. La modularité de l'architecture permet d'envisager plusieurs configurations et donc de s'adapter à différents types d'applications. La gestion de la mémoire, avec en particulier un dispositif de réallocation automatique des cellules de mémoire qui ne sont plus utilisées, est entièrement intégrée dans le matériel et s'effectue en simultanéité totale avec l'exécution proprement dite des programmes.

Quatre processeurs élémentaires sont utilisés (fig. 3), chacun spécialisé dans les tâches de mémorisation, réduction, communication et échange avec l'extérieur, c'est-à-dire entrée et sortie des données. Le processeur de communication sert de brique de base à un réseau à travers lequel les autres processeurs peuvent communiquer entre eux. Ce réseau a aussi une fonction très importante de régulation de la machine entière: il gère la répartition du travail entre les autres processeurs, dirigeant les nouvelles tâches de réduction ou de mémorisation vers les processeurs les moins occupés, contrôlant le degré de parallélisme et répondant par là à notre deuxième question. Par ailleurs, on le voit bien, notre approche consiste à libérer le plus possible le programmeur de cette gestion du parallélisme. Nous répondons à notre quatrième question.

Une maquette reposant sur la technologie dite VLSI (*Very Large Scale Integration*) qui permet une miniaturisation très poussée des processeurs, est prévue pour le courant de l'année 1989 (fig. 4). Déjà le processeur de communication est réalisé. Chaque processeur de réduction pourra, lui, effectuer de l'ordre de deux cent mille réductions par seconde, et la maquette en contiendra seize (fig. 5).

Encore un long chemin.

Si une maquette de MaRS doit bientôt voir le jour, il restera les nécessaires étapes de validation, c'est-à-dire de vérification de performances et de comportements des programmes sur ce type de machines, avant d'en faire un produit industrialisable. Le but du projet est de vérifier en vraie grandeur l'intérêt du modèle de réduction, les performances qu'on peut en attendre et sa capacité de parallélisation. D'autre part, l'étude suscite déjà d'autres questions. Une architecture du type MaRS permet de faire travailler ensemble au plus une centaine de processeurs de réduction. Est-il possible, et comment, de construire des réseaux plus importants? A l'autre bout de la chaîne, du côté du programmeur, on constate que faute de recherches suffisantes, notre savoir-faire en programmation symbolique parallèle est relativement pauvre: même si la prise en compte du parallélisme dans une machine telle que MaRS est aisée, cela ne dispense pas de la recherche de nouvelles « recettes » de programmation pour exploiter efficacement les possibilités du parallélisme. Les programmeurs étant nourris depuis leur formation d'algorithmique séquentielle, une certaine adaptation est nécessaire.

Par ailleurs, l'évolution de l'informatique devrait nous conduire à présent à concevoir l'architecture des ordinateurs à partir des applications que ces machines devront supporter; alors qu'auparavant, on a souvent pu le constater, une nouvelle génération de machines apparaissait pour corriger les insuffisances décelées sur la génération précédente... Cette spécialisation des architectures est actuellement envisageable compte tenu des progrès de la technologie des circuits intégrés et surtout des outils de conception assistée par ordinateur de tels circuits. En quelques mois, un ingénieur peut concevoir une puce de la complexité d'une carte entière de circuits des années 1980, et dont le coût de réalisation d'un prototype est équivalent. Par exemple, en conception ASIC (*Application Specific Integrated Circuits*), c'est-à-dire pour des circuits spécialisés, le coût moyen de fabrication d'un circuit de cinquante mille transistors en technologie dite CMOS, qui permet une grande miniaturisation tout en consommant peu d'énergie, est actuellement de cent mille francs pour une vingtaine d'exemplaires.

L'exemple du projet MaRS est donc caractéristique de cette nouvelle tendance: partant du domaine du traitement symbolique, nous sommes parvenus à la définition d'une machine fonctionnant sur le modèle de la réduction parallèle de graphes, qui est spécifiquement adaptée à ce domaine et dont nous attendons des performances significatives. ■

Pour en savoir plus

Sur la programmation fonctionnelle:

- H. Glaser, C. Hankin, D. Till, *Principes de programmation fonctionnelle*, Masson, 1987.
- P. Henderson, *Functional programming. Application and implementation*, Prentice Hall, 1980.
- S.L.P. Jones, *The implementation of functional programming languages*, Prentice Hall, 1987.
- S. Vegdahl, « A survey of proposed architectures for the execution of functional languages », *IEEE Transactions on Computers, C33*, N° 12, 1984.

Sur le projet MaRS:

- M. Castan, « Mécanismes de base pour la réduction parallèle », *Bigre+Globule, 50*, Groplan 86, septembre 1986, p. 14.
- M. Castan *et al.*, « Le processeur de réduction de MaRS, machine à réduction symbolique », *Bigre+Globule, 56*, C3, novembre 1987, p. 18.
- M. Lemaître *et al.*, « Mechanisms for efficient multiprocessor combinator reduction », *Proc. of the 1986 ACM Conf. on Lisp and functional programming*, Cambridge, Massachusetts, août 1986, p. 113.

NOUVELLE VERSION

STATISTIQUES ET ANALYSE DE DONNEES

FAITES PARLER VOS CHIFFRES !

- **RECHERCHE** appliquée ou fondamentale - Médicale - Biologie - Agronomie - Expérimentation.
- **ÉTUDES** de marché - Enquêtes - Sociologie - Politique - Marketing.
- **INDUSTRIE** Contrôle qualité - Échantillonnage.

CONVERSATIONNEL : produit entièrement français, simple à mettre en œuvre sans aucune connaissance de l'informatique.

MODULAIRE : 6 modules à la carte.
- Gestion de fichiers - Statistiques descriptives - Croisement de variables et X^2 - Items calculés...
- Tests paramétriques : analyse de la variance, t de Student, t sur séries appariées, corrélation, régression linéaire.
- Tests non paramétriques : T de Wilcoxon, r de Spearman, U de Mann-Whitney, X^2 de Friedmann, H de Kruskal-Wallis, Q de Cochran, Mac Nemar.
- Plans expérimentaux : analyse de variance 1, 2 ou 3 facteurs avec ou sans mesures répétées, carrés latins, Duncan, Newman-Keuls, essai croisé.
- Analyse multivariée : ACP, régression multiple, segmentation, analyses discriminante et des correspondances.
- Graphiques : circulaires, histogrammes, nuage de points, courbes.

COMPATIBLE avec tous les grands logiciels du marché : Lotus 1-2-3, dBase, Sylk (Multiplan), Dif, etc.

SUPPORT TECHNIQUE : assistance gratuite, abonnement, mise à jour, fiches techniques, formation.

FORMATION : chaque mois plusieurs stages de formation aux produits et aux statistiques.

IBM et compatibles
Macintosh et Apple II.

P.C.S.M.

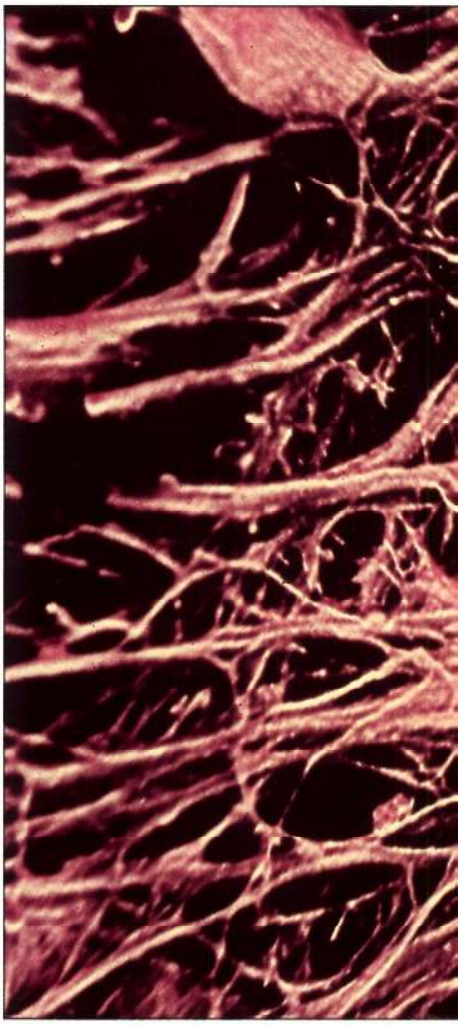
Programme Conversationnel de statistiques pour les Sciences et le Marketing.

DELTASOFT
43, chemin du Vieux-Chêne
ZIRST Grenoble-Meylan
38240 MEYLAN
Tél. 76 41 85 08
Télex 980 610
Télécopie 76 41 83 96

OPTIMA
8, rue Yves-du-Manoir
33700 MÉRIGNAC
(BORDEAUX)
Tél. 56 48 51 62

Les machines neuronales

par Léon Personnaz, Gérard Dreyfus et Isabelle Guyon



Reconnaître des caractères manuscrits est chose aisée pour un jeune enfant... et presque impossible pour un ordinateur. Mais pourrait-il exister un jour des machines « douées » de sens — la vue ou l'ouïe — comparables aux nôtres? Autrement dit, est-il possible de construire des circuits électroniques imitant les capacités exceptionnelles dont font preuve les neurones, organisés en réseaux, dont est fait notre cerveau? Depuis quelques années, des chercheurs de par le monde relèvent ce défi: on assiste à un foisonnement de travaux sur ce qu'il est convenu d'appeler des réseaux de neurones formels, ou des machines neuronales. Des résultats spectaculaires ont déjà été obtenus sur les capacités potentielles de ce type de machines. Nous vous invitons à les découvrir ici. Et si les machines neuronales sont, encore aujourd'hui, largement à l'état de concepts, la micro-électronique permet désormais d'en construire des prototypes.

Nous construisons une machine qui sera fière de nous », tel est le slogan de Thinking Machines Inc. Cette firme américaine commercialise, sous le nom de Connection Machine, un ordinateur qui fait preuve d'une impressionnante puissance de calcul (voir « L'architecture des nouveaux ordinateurs », dans ce numéro). Conformément aux principes de la publicité, la formule frappe juste car elle évoque un mythe ancien et tenace: celui de la machine intelligente. Et pourtant, en dépit de son architecture très originale, dite massivement parallèle — à base de 65 536 calculateurs élémentaires! —, la Connection Machine est un ordinateur au sens habituel du terme. Elle exécute un programme écrit par un utilisateur et stocké dans une mémoire bien localisée. Par conséquent, aussi impressionnantes que soient ses performances, cette machine n'est jamais qu'une calculatrice surpuissante.

Bien sûr, les ordinateurs parallèles se sont révélés extrêmement efficaces pour effectuer des calculs à grande vitesse. Mais ils sont beaucoup moins bien adaptés que le cerveau humain pour résoudre certains types de problèmes: reconnaître des symboles manuscrits est une tâche aisée pour un jeune enfant, mais seuls les ordinateurs les plus puissants réussissent à obtenir des performances acceptables, et ce au prix de l'écriture de programmes complexes par l'utilisateur. Pourquoi une telle différence? Parce que dans le cerveau, les informations sont traitées de manière complètement différente.

En particulier, notre mémoire n'y est pas localisée à un endroit précis. Et la notion même de programme n'a pas de sens! Nos connaissances se mettent en place à travers un processus bien différent : l'apprentissage par modification des interactions entre neurones. Par ailleurs, il est bien connu que le dysfonctionnement d'un certain nombre de neurones — nous en perdons quelques centaines de milliers par jour — n'entraîne pas la mise hors d'usage du système nerveux, ce qui suggère que le cerveau est doté d'une très grande redondance. Cela n'est pas, en général, le cas des ordinateurs...

Les capacités d'apprentissage du cerveau, ainsi que sa résistance aux pannes locales, résultent essentiellement du fonctionnement collectif et simultané des neurones qui le composent, organisés en réseaux fortement interconnectés (fig. 1). Par conséquent, il est tentant de penser qu'un réseau de neurones artificiels, même extrêmement simplifié, pourrait constituer une machine parallèle susceptible de mimer certaines des capacités du cerveau.

La notion de neurones artificiels — on dit aussi « formels » — et l'idée d'organiser ceux-ci en réseaux ne sont pas récentes. Mais ce n'est que depuis le début des années 1980 que les recherches sur ce type de machines se sont multipliées et ont abouti à de nombreux résultats exploitables. Aujourd'hui, des applications pratiques sont envisagées industriellement et des prototypes de machines neuronales existent. C'est ce domaine en pleine effervescence que nous allons parcourir.

Des machines qui apprennent?

Mais avant de nous aventurer plus loin, plusieurs mises au point s'imposent. Tout d'abord, il faut garder à l'esprit que les machines neuronales sont encore largement à l'état de modèles théoriques et qu'il existe très peu de réalisations de telles machines. C'est pourquoi la plupart

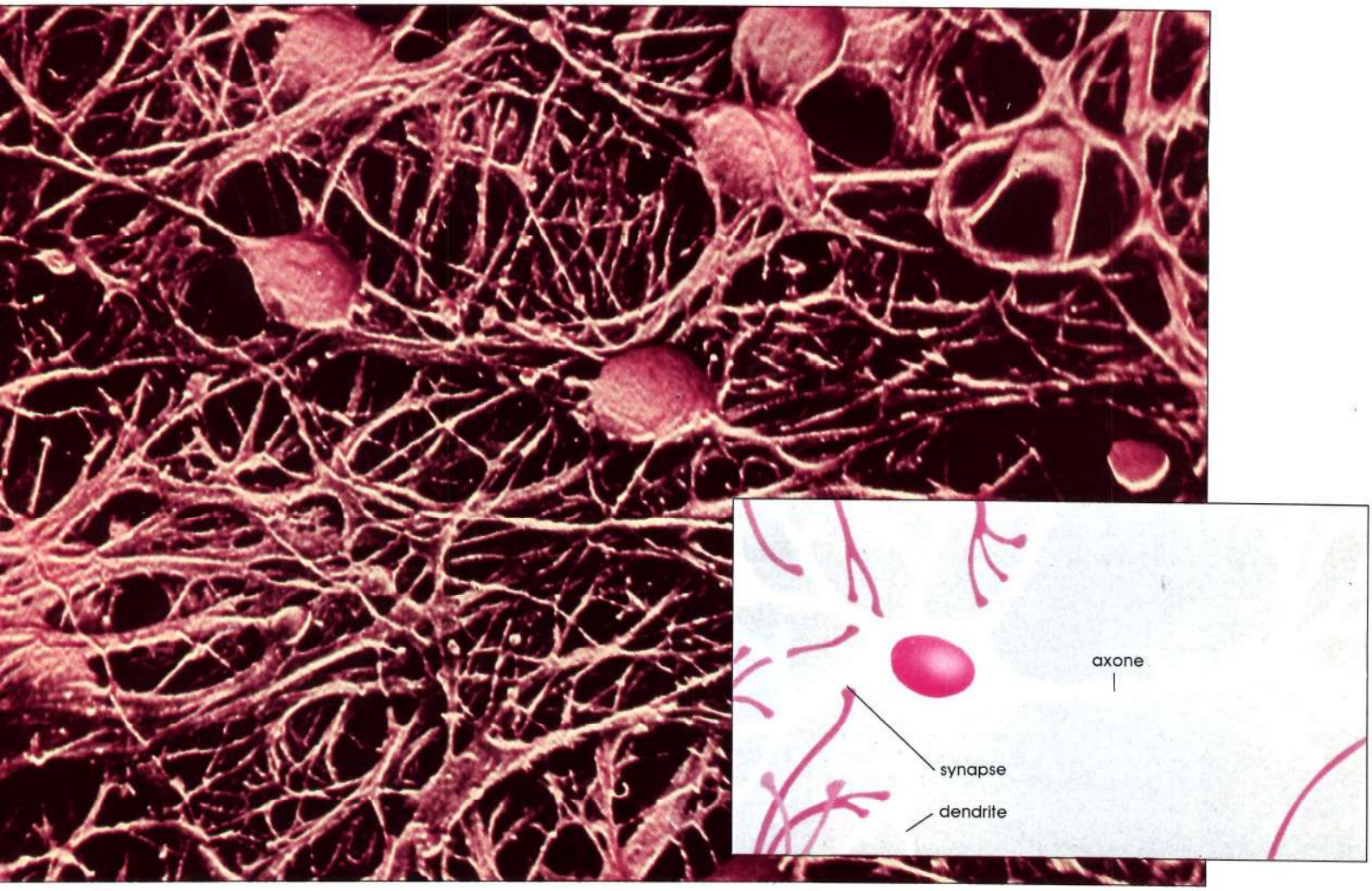


Figure 1. Les « machines neuronales » devraient parvenir à imiter très modestement quelques-uns des talents de notre cerveau. L'activité collective et simultanée de milliards de neurones, ou cellules nerveuses (photographie), confère à notre cerveau des capacités inaccessibles aux ordinateurs classiques, même les plus puissants. Pourtant, chaque neurone réalise, schématiquement, une opération très simple (médaillon) : en fonction des signaux qu'il reçoit des autres neurones par ses dendrites (en bas), il décide d'envoyer ou non un signal à ses congénères le long de son axone (à droite). La communication entre axone et dendrites s'effectue en des points appelés synapses. (Cliché CNRI)

des résultats de simulation que nous allons présenter ici ont été obtenus sur des ordinateurs conventionnels, du type de ceux que beaucoup d'entre nous utilisent quotidiennement. Bien sûr, ces ordinateurs ne sont en général pas eux-mêmes des machines parallèles : néanmoins, ces simulations donnent une idée tout à fait précise des performances potentielles des machines neuronales !

D'autre part, le lecteur est en droit de se poser immédiatement des questions fondamentales. Par exemple, quel genre de tâches peut-on espérer faire accomplir à un réseau de neurones formels ? Plus encore, comment met-on en œuvre, dans une telle machine, un apprentissage ?

Nous pouvons d'ores et déjà fournir une réponse partielle à la première question : certains des réseaux à l'étude constituent ce que les spécialistes ont pris coutume d'appeler des mémoires associatives. Ce sont des machines qui, ayant mémorisé un certain nombre d'informations, sont ensuite capables d'identifier des informations inconnues, mais semblables, et ce par association d'idées, en quelque sorte. Pour reprendre l'exemple que nous évoquions plus haut, les mémoires associatives peuvent donc servir dans la reconnaissance automatique de caractères manuscrits, même lorsque ceux-ci sont mal écrits. Mais les machines neuronales peuvent aussi réaliser d'autres tâches. Nous y reviendrons.

En attendant, comment les informations se mettent-elles en place dans les mémoires associatives ? Justement : au cours d'une phase d'apprentissage. Cela nous ramène donc à la seconde question évoquée ci-dessus : qu'est-ce que l'apprentissage ? En fait, comme nous allons le voir, il s'agit là d'un problème central en matière de réseaux de neurones formels : leur trouver des règles d'apprentissage performantes. Nous allons tenter, tout au long de cet article, de préciser ce concept et de montrer comment les différentes machines neuronales s'acquittent des tâches que nous leur demandons.

Pour comprendre le fonctionnement des neurones formels, examinons brièvement celui des neurones biologiques. Chacun de ces derniers reçoit des signaux provenant de plusieurs milliers de ses semblables par des ramifications appelées dendrites. A partir de ces informations, il décide d'émettre ou non un signal, ce qu'il fait le long d'une fibre unique ramifiée à son extrémité, l'axone. L'information passe d'un neurone à un autre en des points de contact appelés synapses (fig. 1).

Sans entrer ici dans les détails du mode de transmission neurobiologique de l'information, disons simplement qu'une synapse reliant deux neurones peut être de nature excitatrice ou inhibitrice. Dans le premier cas, le neurone émetteur aura tendance à activer le neurone récepteur, et dans l'autre cas le neurone émetteur aura tendance à inhiber l'activité du neurone récepteur. Chaque synapse est en outre caractérisée par l'efficacité avec laquelle elle assure la connexion. En définitive, si le neurone prend bien des décisions en fonction de la somme

Léon Personnaz et **Gérard Dreyfus** sont docteurs en physique. **Isabelle Guyon** est ingénieur. Tous trois sont chercheurs (L. Personnaz et G. Dreyfus sont également enseignants) à l'ESPCI de Paris.

d'informations qu'il reçoit, la contribution de chacune de ces informations est pondérée par l'efficacité de la synapse correspondante.

Le neurone formel est un modèle du neurone biologique qui possède les propriétés que nous venons d'évoquer. Il a été conçu par W. Mc Culloch et W. Pitts, de l'université de Chicago en 1943. C'est un élément binaire dont l'état est +1 (actif) ou — 1 (inactif). Il actualise son état périodiquement de la manière suivante : il calcule la somme de ses entrées (qui sont les sorties des neurones formels auxquels il est relié), la valeur de chaque entrée étant modulée par l'efficacité synaptique correspondante, et prend une décision en comparant cette somme à un seuil qui lui est propre (fig. 2). Si la somme est supérieure au seuil, le neurone se met dans son état actif (+1) ; dans le cas contraire, il se met dans son état inactif (−1). Tous les neurones prennent donc leurs décisions simultanément en tenant compte de l'évolution de l'état global du réseau.

La simulation d'un tel réseau est très simple. Chaque neurone peut être représenté, dans la mémoire d'un ordinateur conventionnel, par un certain nombre de valeurs numériques : son état à un instant donné, les efficacités synaptiques qui le concernent et la valeur de son seuil interne. La simulation du rôle de chacun des neurones constitue un calcul tout à fait élémentaire.

Comment s'assurer que le réseau mémorise, puis sache reconnaître, précisément ce qu'on veut lui faire reconnaître ? Plus généralement, comment s'assurer que le réseau remplisse correctement la tâche qui lui est assignée ? Le neurophysiologiste D.O. Hebb, de l'université Mc Gill de Montréal, avait émis l'hypothèse, dès 1949, que les facultés des réseaux de neurones biologiques sont dues à l'auto-organisation de leurs connexions sous l'effet des stimuli qu'ils reçoivent. Le principe de cette auto-organisation est simple : l'efficacité d'une synapse se renforce lorsque les neurones qu'elle relie ont tendance à être actifs ou inactifs en même temps ; dans le cas contraire elle s'atténue.

Nous avons dit plus haut que le problème crucial consiste à trouver des règles d'apprentissage permettant aux réseaux de neurones formels de remplir une tâche bien définie, par exemple de mimer la capacité de mémoire associative dont fait preuve notre cerveau. Nous pouvons à présent préciser notre propos : trouver une règle d'apprentissage, c'est mettre au point une procédure — c'est-à-dire des instructions explicites —, permettant de calculer les efficacités synaptiques. Le principe qualitatif énoncé par Hebb est un exemple très simple d'un tel calcul, mais il est loin d'être le seul ou le plus efficace.

Premiers réseaux, premières règles d'apprentissage.

Le réseau de neurones formels le plus simple est historiquement le premier. Il comporte des neurones d'entrée ou récepteurs. Ces neurones n'effectuent aucun traitement, leur rôle étant simplement de recevoir et de transmettre l'information à mémoriser. Leur état conserve donc la même valeur tant que rien ne vient le modifier de l'extérieur. Le réseau contient, bien sûr, une couche de neurones capables de calculer leur propre état en fonction des informations provenant des neurones d'entrée ; ce sont ceux qui contribuent véritablement à la décision du réseau. Un tel réseau est donc constitué de deux « couches » de neurones différents. Les neurones de la deuxième couche reçoivent des informations de tous les neurones de la première couche, mais ne sont pas connectés entre eux (fig. 3A).

Comment confère-t-on à un tel réseau la capacité de mémoire associative ? Pour le comprendre, supposons que le réseau soit destiné à la reconnaissance de chiffres manuscrits. Supposons donc, par exemple, qu'il comporte, outre les neurones récepteurs, dix neurones spécialisés chacun dans la reconnaissance d'un chiffre entre 0 et 9. Autrement dit, si l'on numérote ces neurones de 0 à 9, seul le neurone numéro 0 sera actif si le chiffre présenté aux neurones est un zéro, seul le neurone numéro 1 sera actif si le chiffre présenté est un 1, et ainsi de suite. Pour présenter l'image d'un chiffre manuscrit au réseau, il faut commencer par la traduire sous une forme numérique, compatible avec le mode d'expression des neurones : des — 1 et des +1. La façon la plus simple de procéder consiste à découper l'image en petits carrés. A chacun de ces petits carrés, on fait correspondre un neurone récepteur qui prend la valeur +1 si le carré contient une portion du chiffre manuscrit et — 1 s'il est vide.

Les dix neurones calculent alors leur propre état d'après ces entrées, les efficacités des synapses qui les relient aux entrées et leur seuil interne ; la configuration obtenue sur ces dix neurones constitue la réponse du réseau. Mais sans autre précaution, cette réponse n'aura probablement rien à voir avec celle que l'on souhaite obtenir... Comment calculer les coefficients synaptiques pour obtenir la réponse souhaitée ?

Dans les années 1950-1960, les pionniers des machines neuronales avaient conçu deux règles d'apprentissage adaptées au réseau tout simple que nous venons de décrire. L'une d'elles, due à B. Widrow et M. Hoff de l'université de Stanford est appelée aujourd'hui règle de Widrow-Hoff. L'autre, due à F. Rosenblatt, qui travaillait à l'université Cornell, est appelée règle du Perceptron (fig. 4). Ces deux règles permettent de soumettre le réseau décrit ci-dessus à ce qu'on appelle un apprentissage supervisé. Voici en quoi il consiste. On présente aux neurones d'entrée les chiffres à mémoriser, les uns après les autres. Pour chacun des chiffres, on observe la réponse des dix neurones qui effectuent les calculs. Et, comme on connaît *a priori* la réponse que l'on veut obtenir, on essaye de réduire la différence qui existe entre celle-ci et la réponse fournie par le réseau. Comment ? Dans le cas de la règle du Perceptron, en agissant sur les coefficients synaptiques des neurones qui fournissent une réponse

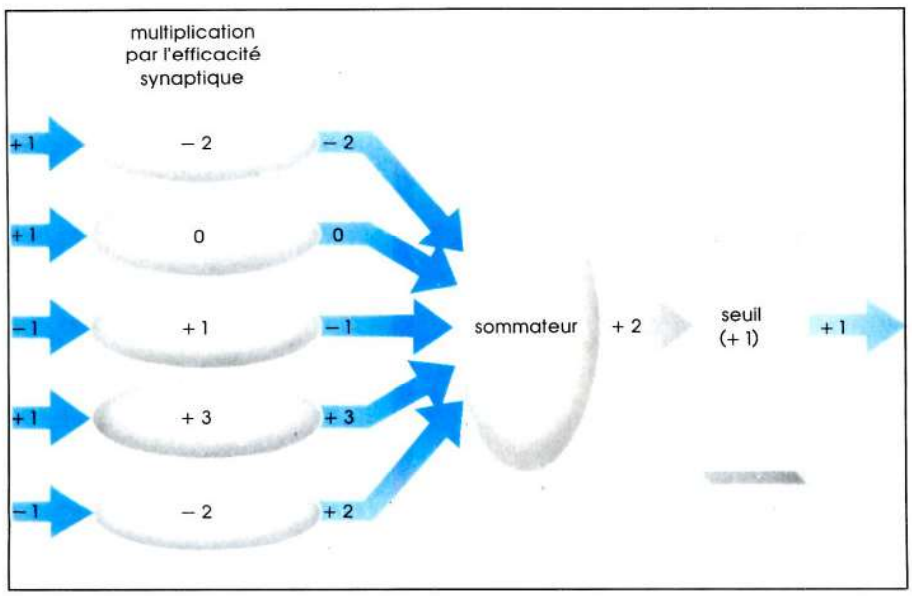


Figure 2. Le modèle de neurone formel utilisé aujourd'hui dans toutes les études de machines neuronales date des années 1940. Il est caractérisé par son état — actif (+1) ou inactif (−1) —, par un seuil interne et par l'efficacité de ses interactions avec les autres neurones formels au niveau de ses synapses. Ce schéma montre comment le neurone formel recalcule son état à chaque instant, en fonction de l'influence globale du réseau. Il multiplie la valeur de l'état de chacun de ses semblables (à gauche) par l'efficacité synaptique correspondante, et additionne le tout (sommateur). Enfin, il compare le résultat à son seuil et en déduit son nouvel état : +1 si la somme est supérieure au seuil, −1 sinon (+1 dans ce cas). Les réseaux de neurones formels sont faciles à simuler sur des ordinateurs classiques (et même sur des micro-ordinateurs).

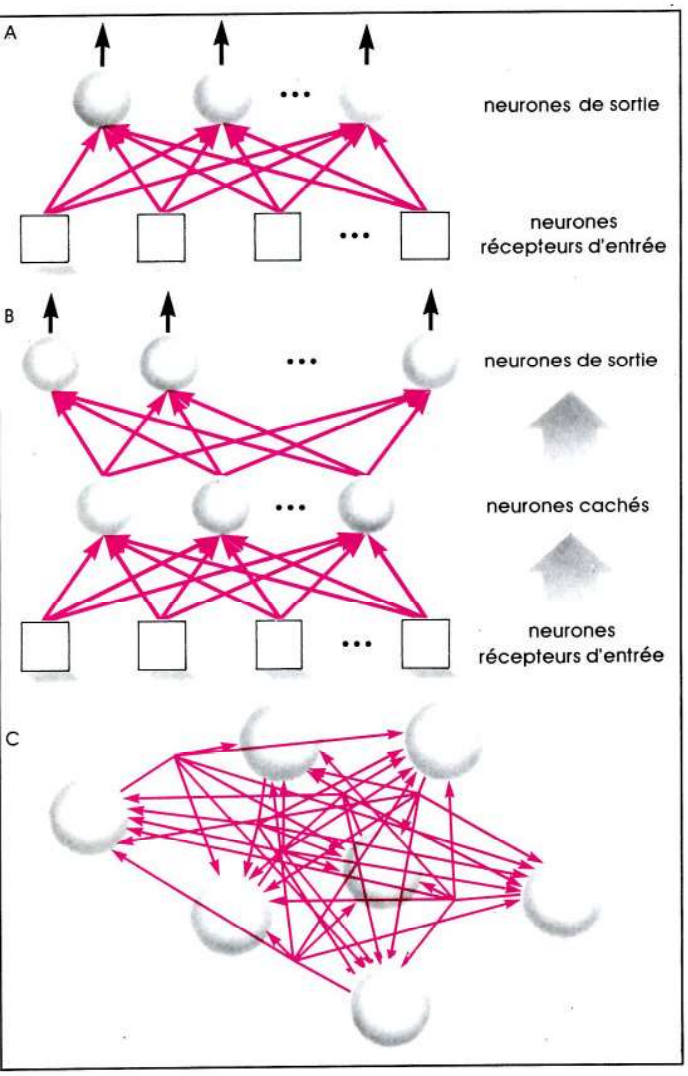


Figure 3. Le réseau de neurones le plus simple comporte deux couches de neurones formels, comme on le voit en A. Des neurones d'entrée reçoivent les informations à identifier (en bas) et d'autres neurones (en haut) effectuent les calculs et fournissent la réponse du réseau. Il existe également des réseaux multicouches, dans lesquels les neurones effectuant les calculs ne sont pas tous directement reliés à l'extérieur; les neurones représentés au centre en B sont des neurones « cachés ». Comme précédemment, il n'existe aucune connexion entre les neurones d'une même couche. Hormis les réseaux à couches, il existe des réseaux dits totalement interconnectés, ou réseaux de Hopfield (C). Les neurones y jouent à la fois le rôle d'entrée et de sortie. Ces différents types de réseaux — à couches et totalement interconnectés — ont des applications particulièrement intéressantes en reconnaissance des formes.

sage ne représentent qu'une infime partie de toutes les formes possibles. Mais, grâce aux exemples qu'il a mémorisés, le réseau est tout de même capable d'en classer correctement un certain nombre qu'il n'a jamais « vues ».

Quoi qu'il en soit, ces réseaux ne pouvaient résoudre que des problèmes simples de classification. Pour des problèmes plus complexes, une solution consiste à organiser la décision en plusieurs étapes, ce qui revient à utiliser un réseau à plusieurs couches (fig. 3B).

Une architecture plus élaborée: les réseaux multicouches.

On observe, d'ailleurs, que les traitements effectués par le système nerveux mettent en œuvre des architectures complexes. Les systèmes sensoriels tels que le système de la vision sont organisés en plus de deux couches: une couche de neurones récepteurs, dans la rétine par exemple, une couche de neurones moteurs, reliés aux muscles et, entre les deux, des couches de traitements intermédiaires comportant des « unités cachées », c'est-à-dire des neurones non reliés directement avec l'extérieur. Cependant, même si les pionniers avaient songé à de telles architectures « multicouches » celles-ci étaient restées inutilisables car on ne connaissait pas de règle d'apprentissage adaptée. En effet, l'utilisation directe des règles d'apprentissage du Perceptron ou de Widrow-Hoff pour calculer les efficacités synaptiques était impossible à cause de l'existence des unités cachées. Puis, en 1974, une généralisation de la règle de Widrow-Hoff connue

incorrecte; dans le cas de la règle de Widrow-Hoff en modifiant les coefficients synaptiques de tous les neurones, proportionnellement à la différence entre la réponse obtenue et la réponse désirée (algorithme de gradient).

Dans des cas simples, au bout d'un certain nombre de présentations de toutes les configurations à mémoriser, et après modifications correspondantes des efficacités synaptiques, celles-ci se stabilisent. On dit que l'algorithme, c'est-à-dire la méthode de calcul utilisée, « converge ». La phase d'apprentissage est achevée. Les efficacités synaptiques ont alors des valeurs optimales au regard des configurations mémorisées.

Au cours de la phase de reconnaissance, on impose aux neurones d'entrée une configuration inédite: un chiffre écrit de la main d'un inconnu, par exemple. Les dix neurones calculent alors leur nouvel état, en fonction des efficacités synaptiques désormais fixées. Deux cas peuvent se présenter: si un seul des dix neurones est actif, le réseau a reconnu un chiffre; si aucun d'entre eux n'est actif, ou si plusieurs le sont, le réseau avoue son impuissance à reconnaître un chiffre. Naturellement, les formes qui lui sont devenues familières au cours de l'apprentis-

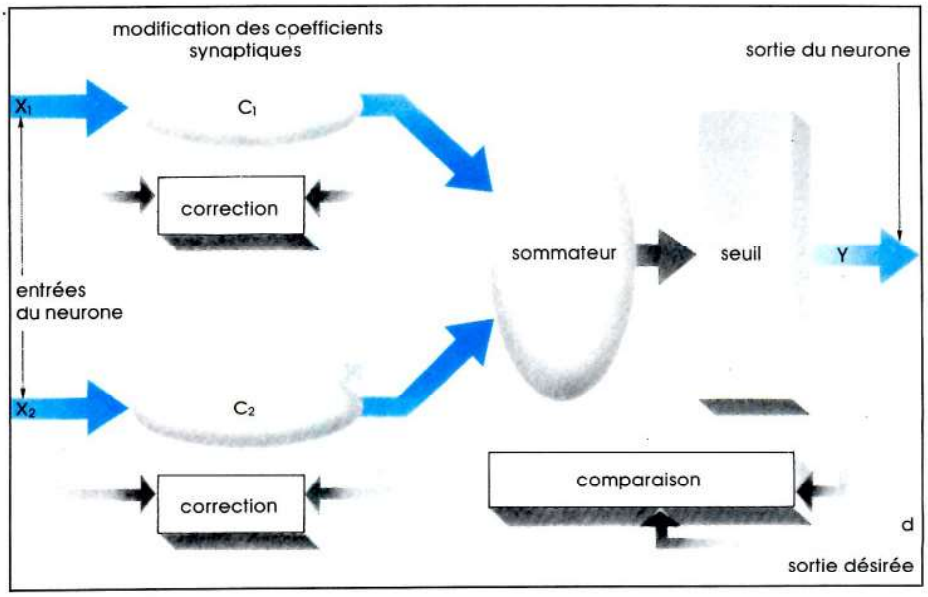


Figure 4. Quel que soit le réseau, les neurones apprennent indépendamment les uns des autres en modifiant pas à pas leurs coefficients synaptiques. Voici un exemple de règle d'apprentissage, celle dite du Perceptron, pour un neurone (dont les éléments sont représentés en gris) ne possédant que deux entrées (et donc deux coefficients synaptiques C_1 et C_2). A chaque présentation d'un nouvel exemple à apprendre (appelé ici x_1, x_2) le neurone calcule sa sortie y (en haut à droite). Si y est égal à la sortie désirée d (en bas à droite), aucune modification n'est effectuée. En revanche, si y est différent de d, on ajoute $d.x_1$ à C_1 et $d.x_2$ à C_2. Cette modification entraîne une diminution de l'écart entre la sortie observée (y) et la sortie désirée (d). Au bout d'un certain nombre de présentations de tous les exemples, le neurone répond correctement: il a « mémorisé » les exemples.

sous le nom de méthode de « rétro-propagation du gradient » a permis de surmonter cet obstacle.

Sans entrer dans des détails techniques, disons simplement qu'elle consiste, elle aussi, à imposer une configuration aux neurones d'entrée, à observer la réponse du réseau, fournie par les neurones de sortie, puis à recalculer les efficacités synaptiques de façon à minimiser l'écart qui sépare la réponse réelle de la réponse souhaitée par une méthode de gradient. Mais ici, le calcul se fait couche par couche, de la sortie vers l'entrée (d'où le nom de rétro-propagation).

L'un des exemples d'application les plus spectaculaires de cette méthode d'apprentissage est le système « Nettalk », un réseau de neurones qui apprend à lire à haute voix une page de texte en anglais, sans qu'on lui enseigne explicitement les règles de prononciation!

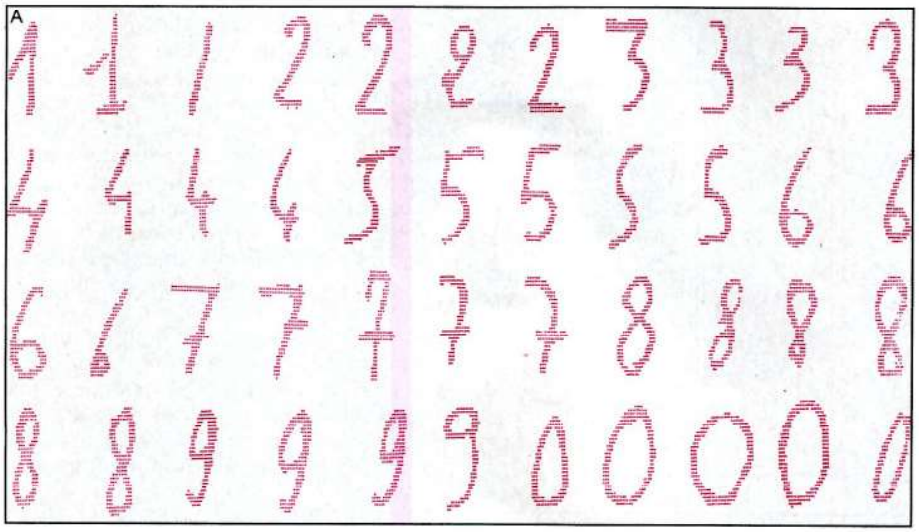


Figure 5. Dans notre laboratoire de l'ESPCI, à Paris, nous avons simulé, sur un dispositif expérimental, un réseau totalement interconnecté de neurones formels qui apprend à reconnaître des chiffres manuscrits. Ce réseau comporte six cents neurones disposés sur un tableau rectangulaire. Sur le schéma A on voit les prototypes, correspondant à des styles d'écriture différents, qui ont été mémorisés par le réseau pour chacun des dix chiffres (les points représentent les neurones « actifs », c'est-à-dire ceux dont l'état est +1). Lorsque l'on présente aux six cents neurones de la mémoire un chiffre écrit dans un style inconnu, ils recalculent tous simultanément leurs états respectifs. Au bout d'un certain nombre d'étapes, l'état global du réseau se stabilise: celui-ci a reconnu le chiffre. Pour se représenter l'évolution du réseau, on peut imaginer une boule de billard qui parcourt un paysage vallonné, descend vers le fond du vallon le plus proche et y reste (B). L'expérience prouve que notre réseau répond correctement dans 80 % des cas, n'arrive pas à décider dans 10 % des cas et se trompe dans les 10 % restants.

Conçu en 1985 par T. Sejnowski, de l'université John Hopkins de Baltimore et C.R. Rosenberg de Princeton, il a été simulé sur un ordinateur classique muni d'un dispositif de synthèse de la parole. Ce réseau comporte trois couches: deux cent trois neurones d'entrée, répartis en sept groupes de vingt-neuf, vingt-six neurones de sortie pour chacun des vingt-six phonèmes possibles, et quatre-vingt neurones cachés dans la couche intermédiaire[1].

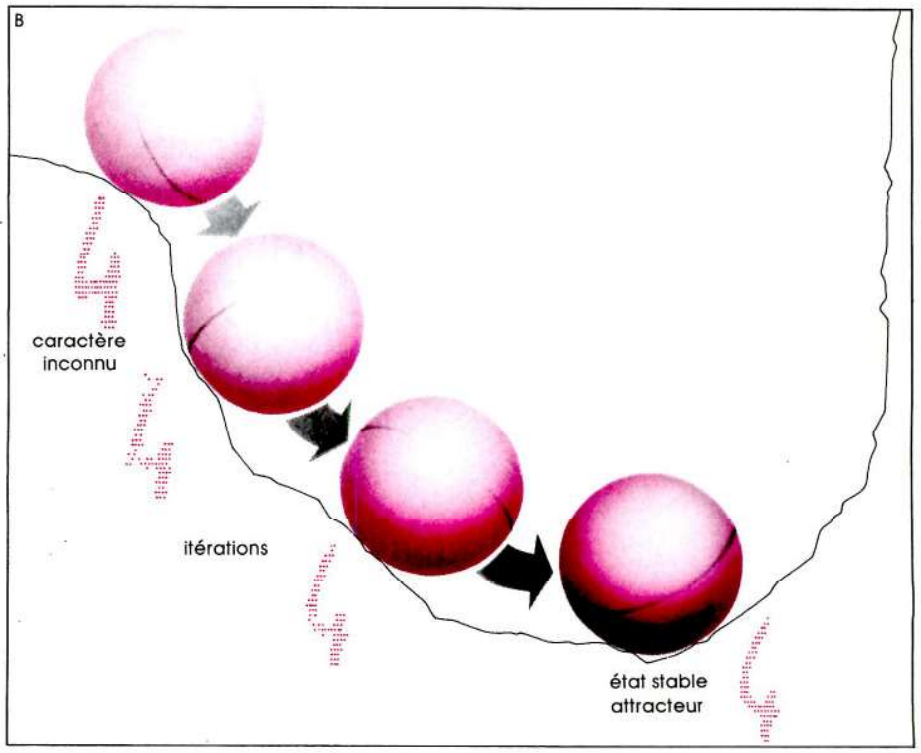
Chacun des sept groupes de neurones d'entrée reçoit le code d'un caractère alphabétique: le réseau voit donc des groupes de sept caractères, mais de façon telle que tout se passe comme s'il « lisait » à travers une fenêtre glissante. Cette procédure est facile à justifier. En effet, l'apprentissage est effectué de la façon suivante: on présente à l'entrée du réseau un groupe de sept caractères du texte, et l'on calcule les efficacités synaptiques de manière à obtenir sur la couche de sortie une réponse aussi proche que possible du code du phonème correspondant à la lettre centrale de ce groupe, les six autres lettres fournissant un contexte dont dépend la prononciation de la lettre centrale. On déplace ensuite la fenêtre d'un caractère vers la droite et l'on répète le processus. On continue ainsi jusqu'à ce que toute la page ait été présentée. Bien entendu, il est nécessaire de présenter le même texte un très grand nombre de fois pour que le calcul des efficacités synaptiques converge vers une solution.

Lorsqu'une page a été apprise de cette manière, on présente au réseau une page inconnue. Celui-ci est alors capable, dans une certaine mesure, de « deviner » la prononciation de mots qui ne figuraient pas dans la page ayant servi à l'apprentissage. Les expériences effectuées par T. Sejnowski et C. Rosenberg montrent que le réseau possède de surprenantes capacités de généralisation.

L'algorithme d'apprentissage par rétro-propagation connaît à présent des améliorations et des variantes. Cependant, certains problèmes pratiques posés par les réseaux multicouches ne sont pas encore résolus, comme par exemple le choix du nombre de couches cachées et d'unités par couche. Et surtout, les propriétés de généralisation ne sont pas encore bien maîtrisées. Des questions passionnantes restent donc ouvertes.

Des réseaux dynamiques: les réseaux de Hopfield.

Dans les réseaux à couches que nous venons de décrire, lorsqu'un stimulus à reconnaître est présenté aux neurones d'entrée, la réponse du réseau est calculée instantanément, après la traversée des couches du réseau. Cependant, outre son organisation en couches, le cerveau humain possède une autre caractéristique essentielle: sa structure permet une recirculation de l'information, de telle sorte que les décisions ne sont pas prises instan-

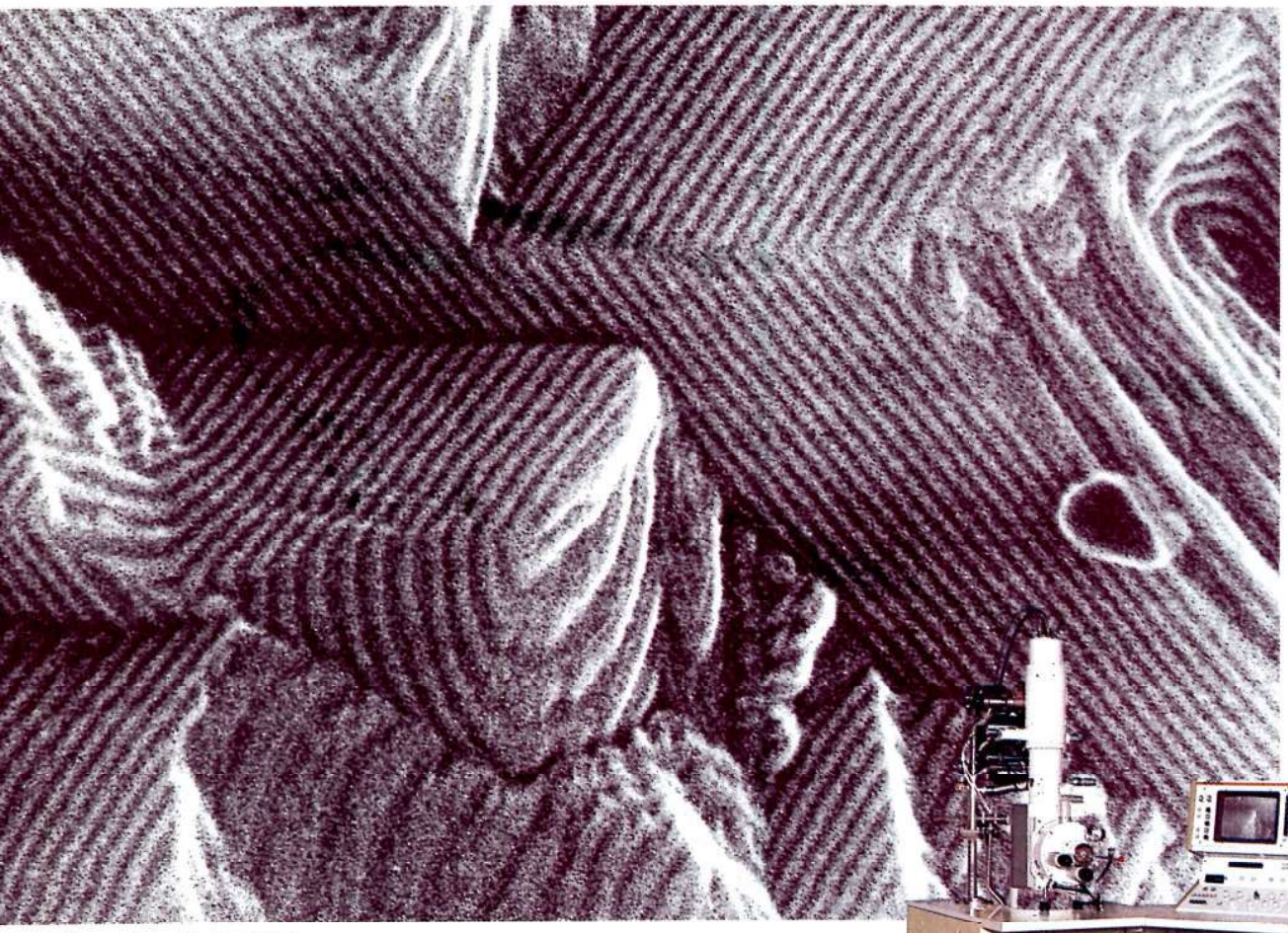


(1) T.J. Sejnowski et C.R. Rosenberg, « NETtalk: a parallel network that learns to read aloud », *Technical Report JHV/EECS – 86/01*, Johns Hopkins University, juin 1986.

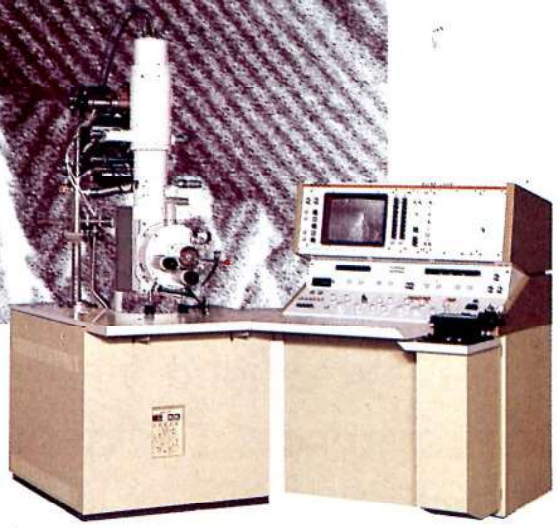
suite page 1369

JSM 840 F*

LE MICROSCOPE A BALAYAGE A EMISSION DE CHAMP



Super réseau Ga As / Al As



LA ULTRA HAUTE RESOLUTION AVEC :

- Canon à cathode froide
- Large gamme de haute tension de 500 V à 40 kV
- Grande chambre objet
- Platine goniométrique eucentrique motorisable, de grande précision
- Sas d'introduction d'objet (100 mm)
- Mise au point automatique
- Optique asservie par microprocesseur

UNE LARGE GAMME D'EQUIPEMENTS PERIPHERIQUES :

- Interface pour contrôle des paramètres optiques par un processeur externe
- Intégration et stockage d'images 1024 x 1024
- Analyse X par dispersion d'énergie
- Analyse d'images...

* Présenté au salon "MESUCORA" du 14 au 18 Novembre 1988 – Allée I – Stand 66

JEOL (EUROPE) S.A.

16, avenue de Colmar – 92500 Rueil-Malmaison
Tél. (1) 47.49.67.00 – Télécopie (1) 47.52.10.41 – Télex 203 337 F

MEDLINE MINITEL

Interrogation guidée, en français ou en anglais, ne nécessitant pas de formation préalable.

A partir d'un Minitel ou d'un micro-ordinateur avec émulation Minitel, accès immédiat à 5 millions de références d'articles scientifiques.

Affichage des références avec ou sans résumé.

Commande immédiate de la photocopie de l'article.

SUR LE KIOSQUE
SANS CONTRAT NI MOT DE PASSE

36.29.00.36

PAR CONTRAT

36.13 INSERM

RENSEIGNEMENTS :
Laissez vos coordonnées sur le 36.13 INSERM
ou adressez-vous à
INSERM-IMA, Hôpital de Bicêtre
78, rue du Général Leclerc 94270 Le Kremlin-Bicêtre
Tél. : (1) 46.71.86.87 poste 312-Télécopie : 46.58.40.57

INSERM
Institut National de la Santé et de la Recherche Médicale
IMA, Hôpital de Bicêtre
94270 Le Kremlin-Bicêtre
Tél. : (1) 46.71.86.87. p. 312

Telesystemes Questel
83-85, boulevard Vincent-Auriol
75013 Paris
Tél. : (1) 45.82.64.64. p. 444

suite de la page 1366

tanément, mais par étapes successives.

Pour illustrer un tel fonctionnement, prenons un exemple : on en conviendra, un objet comportant un plateau fixé à quatre pieds et un dossier évoque, chez tout un chacun, l'image mentale d'une chaise. Toutefois, le processus qui permet d'associer l'image mentale « chaise » au stimulus visuel provoqué par un objet ayant ces caractéristiques n'est certainement pas instantané. A tel point qu'il peut même arriver qu'après un certain temps, l'objet observé, qui avait été pris initialement pour une chaise, soit identifié comme un prie-Dieu...

Force est donc de constater que, contrairement aux réseaux à couches de neurones formels, le cerveau fonctionne apparemment comme un système dynamique, qui n'atteint pas instantanément un état d'équilibre lorsqu'il est soumis à un stimulus extérieur. C'est justement cette propriété qui a inspiré J. Hopfield, du California Institute of Technology, dès 1982[2]. Il a analysé un modèle de réseau dans lequel chaque neurone reçoit des informations de tous les autres neurones du réseau et envoie lui-même des signaux à tous les autres neurones (fig. 3C) : il s'agit donc d'un système « coopératif », dont la décision est prise par étapes successives. Comme nous allons le montrer, la caractéristique essentielle de son comportement réside dans l'existence d'états stables dits « attracteurs ».

Pour illustrer les possibilités des réseaux de Hopfield, nous avons utilisé un dispositif expérimental mis au point au laboratoire d'électronique de l'ESPCI (Ecole supérieure de physique et de chimie industrielles), destiné à la reconnaissance d'un code postal manuscrit (fig. 5). Un module électronique assure l'enregistrement et la séparation des cinq caractères du code postal, qui défilent devant une caméra. Chaque forme de caractère, une fois numérisée, est représentée par un tableau de six cents cases ou pixels, binaires. Chacun de ces pixels est associé à un neurone ; l'ensemble constitue un réseau totalement interconnecté. Les différences de style d'écriture des caractères nous ont conduits à faire mémoriser au réseau plusieurs prototypes par chiffre (fig. 5A). Pendant la phase d'apprentissage, on impose un prototype à mémoriser aux six cents neurones. A chaque fois, on laisse les six cents neurones ajuster leurs synapses, à l'aide d'une règle de type Widrow-Hoff, de telle manière que les prototypes soient des états stables attracteurs. Lorsque toutes les configurations ont ainsi été présentées, l'apprentissage est achevé et les efficacités synaptiques fixées. Ensuite, lors de la phase de reconnaissance, on impose aux six cents neurones un caractère inconnu. A partir de cette configuration, les six cents neurones recalculent leur état, et le réseau évolue ainsi, en passant par des états transitoires, vers une configuration stable (fig. 5B). Si le caractère inconnu ressemble à l'un des prototypes, le réseau aboutira à ce prototype. Ainsi, un prototype du chiffre 4 aura la propriété d'attirer à lui les versions incomplètes ou déformées de ce chiffre, d'où son nom d'attracteur. Mais on observe que le réseau peut également aboutir à un état qui n'est pas l'un des prototypes, mais qui, néanmoins, ressemble à l'un d'eux ! Autrement dit, il s'est stabilisé dans un état « parasite », non prévu par l'apprentissage, et représentant un style de chiffre qu'il a en quelque sorte inventé. Un système simple de codage permet d'identifier tous les états stables qui s'apparentent à un même chiffre. L'ensemble des chiffres correctement reconnus représente 80 % des formes présentées. Dans 10 % des cas, le code final n'est pas interprétable ; cela signifie que le réseau refuse de reconnaître une forme qu'il trouve ambiguë. Une telle propriété est particulièrement intéressante pour les systèmes de reconnaissance des formes. Enfin, 10 % des caractères inconnus sont mal classés.

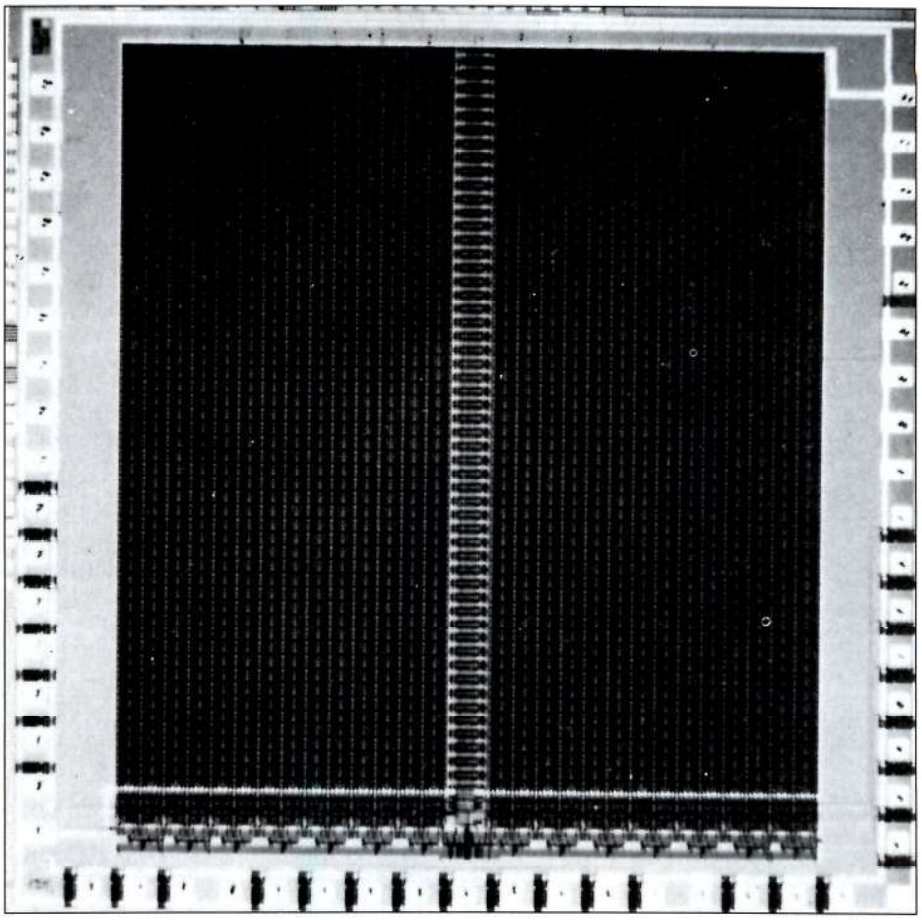


Figure 6. Aujourd'hui, de « vrais » réseaux de neurones sur circuits micro-électroniques existent à l'état de prototypes de taille réduite. Cette photographie représente le réseau électronique en technologie « CMOS » construit par les Laboratoires Bell de la compagnie américaine AT & T en 1986. Il comporte cinquante-quatre neurones tous interconnectés. A quand la première véritable machine neuronale ? (Cliché AT&T Bell Laboratories)

Comment cela marche-t-il ? La réponse vient de la physique, et plus particulièrement de l'étude des systèmes magnétiques qui sont composés d'éléments — ou spins — pouvant prendre deux états magnétiques opposés : appelons-les −1 et +1, comme ceux des neurones formels.

Les interactions entre ces spins sont exactement analogues aux coefficients synaptiques. L'état d'une substance magnétique est caractérisé par ce qu'on appelle son énergie, et les états d'équilibre stable correspondent aux configurations pour lesquelles cette énergie prend des valeurs minimales. Pour se représenter l'évolution spontanée d'un système magnétique, il suffit d'imaginer une bille qui, parcourant un « paysage » vallonné, représentant toutes les configurations possibles de − 1 et de +1 de ses éléments, descend naturellement vers le fond du vallon le plus proche et y reste. Or Hopfield, qui connaissait bien la théorie des systèmes magnétiques désordonnés appelés verres de spin, a justement montré qu'il est possible de décrire mathématiquement les réseaux de neurones formels totalement interconnectés exactement comme s'il s'agissait de verres de spin. L'apprentissage revient alors à « creuser les trous » aux bons endroits dans le paysage d'énergie du réseau ! En effet, le calcul montre que si l'apprentissage est convenablement effectué, la fonction d'énergie d'un tel réseau prend des valeurs minimales exactement pour les configurations qui ont été présentées au réseau durant l'apprentissage. Autrement dit, le réseau a calculé ses efficacités synaptiques de telle sorte

(2) J.J. Hopfield, *Proc. Natl. Acad. Sci., USA, 79,* 2554, 1982.
(3) L. Personnaz *et al.*, *Phys. Rev. A., 34,* 4217, 1986.

Des prototypes de « puces » neuronales existent déjà dans les laboratoires.

que les configurations qu'il doit mémoriser soient des états stables d'énergie minimum[3].

L'exemple que nous venons de présenter entre dans la catégorie des mémoires associatives: il s'agit toujours d'imposer des états stables attracteurs dans l'espace des états du réseau. La propriété essentielle des réseaux de neurones complètement connectés est la recherche rapide (puisqu'elle est faite en parallèle) de minima de la fonction énergie. Cette propriété a également conduit à utiliser ces réseaux pour résoudre d'autres types de problèmes caractérisés par un très grand nombre de solutions parmi lesquelles il faut trouver la « meilleure », ou tout au moins l'une des meilleures, au sens d'un critère qui ne sera autre que l'énergie d'un réseau conçu pour la circonstance. Ce sont des problèmes dits d'optimisation combinatoire. Le problème à résoudre ne consiste plus alors à retrouver un état particulier parmi d'autres, mais à choisir les paramètres d'un réseau tels que la solution du problème soit représentée par le minimum absolu de la fonction énergie[4]. J. Hopfield et D. Tank ont mis sous cette forme le célèbre problème du « voyageur de commerce »: un voyageur de commerce doit faire une tournée en passant par n villes; il doit trouver la tournée la plus courte en ne passant qu'une seule fois par chaque ville. Ce problème, bien que très facile à formuler, est l'un des problèmes d'optimisation combinatoire les plus difficiles à résoudre sur ordinateur, car le temps de calcul nécessaire augmente exponentiellement avec le nombre de villes. Avec un réseau de neurones électroniques, la solution serait trouvée en quelques microsecondes.

Il ne faut pas se méprendre sur la signification de ce travail: il ne s'agit pas, à proprement parler, d'une application (qui se soucie réellement de construire un circuit électronique spécifique pour résoudre ce problème?), mais plutôt d'une démonstration du fait que les réseaux de neurones formels sont à même de résoudre des problèmes qui comptent parmi les plus difficiles.

Mais... où sont donc les neurones formels?

Les résultats tout à fait encourageants obtenus par simulation suggèrent deux directions: la conception de simulateurs spécialisés de réseaux de neurones, et la création de « vrais » réseaux de processeurs parallèles, de conception voisine de celle des neurones formels que nous avons décrits. En effet, c'est seulement avec de tels systèmes que l'on pourra profiter des véritables atouts des réseaux de neurones: la vitesse et la tolérance aux pannes. Le problème principal, pour la réalisation de réseaux, est évidemment celui de la forte connectivité devant exister entre les neurones.

Un véritable marché industriel est en train de s'ouvrir, aux États-Unis, pour les simulateurs de réseaux de neurones. Il s'agit généralement d'ordinateurs personnels munis de cartes spécialisées permettant d'effectuer certaines opérations à grande vitesse. A notre connaissance, trois simulateurs ont été conçus en France dans les laboratoires. L'un a été construit, en 1987, par P. Peretto au CENG (Centre d'études nucléaires de Grenoble), et un autre conçu et réalisé, en 1987, par l'équipe de J. Hérault, de l'IMAG (Institut de mathémtiques appliquées de Grenoble). Enfin une machine est opérationnelle à l'ESPCI. Elle est réalisée à partir de transputers, qui sont des microprocesseurs spécialement conçus pour fonctionner dans des réseaux de processeurs (voir « Les supercalculateurs à transputers », dans ce numéro). Elle a notamment permis d'effectuer des études sur les règles d'apprentissage pour les réseaux de neurones.

Une autre approche pour concevoir des machines de simulation de mémoires associatives consiste à exploiter le parallélisme inhérent aux systèmes optiques, en utilisant ce qu'on appelle des hologrammes (voir « L'ordinateur optique », dans ce numéro). D. Psaltis de l'Institut de technologie de Californie (CALTECH)[5] a construit plusieurs maquettes d'ordinateurs neuronaux dont l'un possède dix mille commutateurs optiques et deux hologrammes plans pour les interconnexions. En France, un projet de calculateur neuronal opto-électronique est en cours d'élaboration depuis 1987 dans le cadre d'une collaboration entre l'ESPCI et le GESSY (Groupe d'études « signaux et systèmes ») de Toulon.

Quant aux premières mémoires associatives sur circuits micro-électroniques, elles ont été conçues et réalisées par les Laboratoires Bell dès 1986[6] (fig. 6). Dans de tels circuits, les neurones sont des objets physiquement bien définis — ce sont des amplificateurs opérationnels utilisés comme des additionneurs — et les efficacités synaptiques sont les valeurs des résistances d'entrée des additionneurs. Ici, les calculs sont réalisés non seulement en parallèle, mais directement sur les courants électriques qui parcourent ces circuits. Autrement dit, ces réseaux sont des calculateurs analogiques, contrairement aux ordinateurs qui sont des calculateurs numériques. Le premier prototype comportait vingt-deux neurones (quatre cent quatre-vingt-quatre résistances) et son temps moyen de réponse était compris entre une et dix microsecondes. En outre, les expériences et les simulations effectuées par les chercheurs des Laboratoires Bell indiquent que les comportements statique et dynamique de leur circuit sont insensibles aux défauts des résistances d'interconnexion, ce qui leur permet d'envisager un haut niveau d'intégration utilisant cette architecture. Un prototype de deux cent cinquante-six neurones, utilisant une technologie légèrement différente, a été construit en

(4) J.J. Hopfield, D.W. Tank, *Biol. Cybern.*, *52*, 141, 1985.
(5) D. Psaltis *et al.*, *Neural Networks Journal*, sous presse.
(6) S. Mackie *et al.*, « *Implementations of Neural Network Models in* Silicon », *in Neural Computer*, R. Eckmiller and C. von der Malsburg (eds), Springer, 1987, p. 467.

1987. En France, un réseau de neurones réalisé sous forme de circuit intégré numérique est à l'étude à l'Ecole polytechnique et à l'ESPCI depuis 1987. Car il ne faut pas oublier que le numérique présente lui aussi des avantages, dus notamment au fait que l'on peut coder les efficacités synaptiques sur un petit nombre de bits — c'est-à-dire en utilisant peu de place dans la mémoire locale de chaque neurone — sans dégradation des performances du réseau. Reste à savoir, bien sûr, qui de l'analogique ou du numérique sera le plus attrayant, aussi bien du point de vue du fabricant que de celui de l'utilisateur.

Au-delà de ces considérations pratiques, on peut se demander si l'intérêt actuel pour les réseaux de neurones formels est une mode passagère, à caractère cyclique, et donc destinée à disparaître aussi soudainement que les Perceptrons des années 1960, ou s'il est le signe d'un prochain bouleversement dans notre manière de concevoir le traitement de l'information. La vérité est probablement entre les deux. En effet, s'il est vrai que l'enthousiasme que l'on constate actuellement aux Etats-Unis est exagéré, les progrès enregistrés en très peu de temps sont le signe que de nouveaux concepts sont effectivement apparus ; de plus, les technologies de la micro-électronique et de l'opto-électronique offrent des possibilités considérables.

Des chercheurs européens ont participé au mouvement dès ses débuts, et la Communauté européenne encourage le développement de ces efforts : sous l'impulsion notamment de G. Toulouse, de l'Ecole normale supérieure, en France, et de C. von der Malsburg, du Max Planck Institut de Göttingen en Allemagne, le projet BRAIN *(Basic Research in Artificial Intelligence and Neurocomputing)* a été lancé en 1987. Il regroupe des équipes de différents pays, dont plusieurs laboratoires français. Bien entendu, les problèmes pratiques très importants que posent encore les réseaux de neurones formels ne seront peut-être pas résolus avant longtemps. Mais l'enjeu justifie largement les efforts déployés. ■

Pour en savoir plus

- D. Rumelhart, J.L. McClelland, *Parallel Distributed Processing*, 2 vol., MIT Press, 1986.
- J.L. Casti, A. Karlqvist (eds), *Real Brains, Artificial Minds*, North-Holland, 1987.
- S. Grossberg, *The adaptative brain*, 2 vol., North-Holland, 1987.
- *IEEE first international conference on neural networks*, 4 vol., San Diego, California, juin 1987.
- R.P. Lippmann, « An introduction to computing with neural nets », *IEEE ASSP Magazine*, *4*, 4, 1987.
- Special section on neural networks for systems and control, *in Control Systems, IEEE Control Systems Society*, *8*, 3, 1988.

- Pour une bibliographie plus complète voir page 1424.

SPECIAL MASSON / LES NOUVEAUX ORDINATEURS

NOUVEAUTÉS

De nouvelles voies vers
l'Intelligence Artificielle
Pluri-disciplinarité
Auto-organisation
Réseaux neuronaux
par J.-C. PEREZ
Préface de R. MOREAU
1988, 248 pages

Etat de l'art, une autre voie pour l'intelligence artificielle. *Les ''grands'' de l'I.A. La voie analytique et formelle. Voie connexionniste et ordinateurs ''neuronaux''.* ***Les racines pluri-disciplinaires de l'I.A.*** *Perturbations, entropie et hasard. Neurobiologie et hologrammes. Dynamique et chaos.* ***De la particule à la métaconnaissance, ''les ordinateurs ondulatoires''. ''Chaleur'',*** *un système d'I.A. issu de la thermodynamique.* ***''Waves'',*** *un système d'I.A. issu de l'holographie.* ***''Chaos fractal'',*** *un système d'I.A. issu du magnétisme. Conclusions et biographie.*

La machine à connexions
par W.D. HILLIS
1988, 200 pages

Cet ouvrage introduit une **architecture d'ordinateurs révolutionnaire,** la machine à connexions, capable d'exécuter des tâches qu'aucune machine conventionnelle séquentielle ne peut entreprendre avec des temps de calcul raisonnables.
Alors que la plupart des ordinateurs ne comportent qu'une unité de traitement et traitent les problèmes de façon séquentielle en enchaînant les calculs les uns à la suite des autres ; **la machine à connexions, avec ses 65 536 processeurs** résout les problèmes ''en parallèle'' : toutes les unités de traitement communiquent et calculent simultanément.

Programmer une machine à connexions.
Considérations de conception.
Le prototype.
Structures de données pour la machine à connexions.
Allocation de mémoire.
Les nouvelles architectures et leur relation avec la physique.

An adaptive neural network : the cerebral cortex
par Y. BURNOD
1988, 376 pages

Cet ouvrage se place volontairement dans une perspective multi-disciplinaire, à l'interface entre la Neurobiologie, la Psychologie, les théories de réseaux de neurones et l'Intelligence Artificielle.

L'auteur propose un modèle de réseaux de neurones pour le Cortex cérébral en restant le plus près possible des connaissances expérimentales.

L'originalité de cette théorie est d'intégrer les quatre principaux niveaux d'organisation du Cortex cérébral (cellules, colonnes corticales, cartes, et systèmes sensori-moteurs), pour montrer comment le tissu cellulaire cortical peut produire les principales fonctions adaptatives étudiées par l'intelligence artificielle : reconnaissance invariante de formes, positionnement dans l'espace, construction de programmes structurés et apprentissage du langage.

A PARAÎTRE PROCHAINEMENT DANS LA COLLECTION
études et recherches en informatique

Algorithmes et architectures systoliques
par Y. ROBERT et P. QUINTON
Cet ouvrage fournit une synthèse sur les architectures systoliques autrement dit les architectures parallèles spécialisées composées d'un grand nombre de processeurs évoluant de manière synchrone.

Aspects mathématiques des réseaux de Pétri
par Ch. REUTENAUER
Les réseaux de Pétri constituent l'un des outils que fournissent les mathématiques appliquées à l'informatique.
Cet ouvrage est plus particulièrement consacré aux problèmes de décidabilité et notamment à la démonstration du théorème sur l'accessibilité.

PARUTIONS DEPUIS 1987

Processus séquentiels communicants
par C.A.R. HOARE
Collection Manuels Informatiques Masson
1987, 288 pages

Calcul de formes par ordinateur
par J. WOODWARK
Collection Manuels Informatiques Masson
Série industrielle
1988, 176 pages

Infographie et applications
par T.M. LIEBLING, H. RÖTHLISBERGER et coll.
Collection Manuels Informatiques Masson
1988, 512 pages

Les systèmes intelligents basés sur la connaissance
par W.J. BLACK
Collection Manuels Informatiques Masson
1988, 192 pages

Génie cognitif
par C. VOGEL
Collection Sciences cognitives, 1988, 200 pages

Acquisition du savoir pour les systèmes experts
par A. HART
Collection Sciences cognitives, 1988, 160 pages

DES LIVRES QUI PARTICIPENT A LA RÉVOLUTION DES NOUVEAUX ORDINATEURS