LES ALGORITHMES PARTICULIERS DE CIME.2

Agnès ETIFIER

INTRODUCTION

Cime (Cartographie Intelligente en Milieu montagnard), est un système expert destiné au traitement d'images satellitaires [MERING 88]. Initialement, les régions étudiées étaient les zones montagneuses du Nepal, le but étant de trouver la meilleure chaîne de traitements pour produire des cartes thématiques de la végétation. CIME.1, la première version de CIME [Monjanel 87], devait faire les preuves de faisabilité de l'implémentation sous forme de système expert d'une démarche en télédétection ayant pour but la cartographie thématique. Ce but a été atteint, et l'expérience a permis de révéler les difficultés inhérentes à une telle implémentation. : un aspect encore très procédural, imposé au système par les règles; une imbrication trop forte entre entre le contrôle (la partie informatique du système), et la stratégie; un objectif limité (on ne peut aller plus loin que la production d'une chaîne de traitements); et une lourdeur, source d'inefficacité, dans la gestion des données utilisées. L'étude précédant la mise en place de CIME.2, la seconde version de CIME [Etifier 88], a mis en avant la nécéssité d'adopter des algorithmes spécialement destinés à la gestion des hypothèses d'une part (ce qui permet à l'expert d'envisager une stratégie particulière, et des objectifs plus ambitieux que la seule production de cartes thématiques), et à la gestion des données utilisées dans le système d'autre part (il s'agit de la gestion de la base de connaissances, et des nécessaires mise à jour qui doivent y être faites). Ce sont donc ces deux algorithmes qui sont présentés ici.

I. CH

I.1. Les hypothèses

Une session de CIME consiste à sélectionner une chaîne de traitements de l'image parmi plusieurs. Ces chaînes étant elles mêmes une succession d'étapes, une étape est l'application d'un traitement à l'image suivie de l'évaluation des résultats obtenus. La meilleure des chaînes est celle pour laquelle **le choix** des traitements et de leurs paramètres est pertinent. Puisque plusieurs combinaisons sont possibles (l'ordre d'introduction de ces paramètres étant important), les choix à effectuer doivent être supervisés. De plus, dans le cas où plusieurs méthodes sont envisageables pour produire les éléments nécessaires au traitement, le nombre de possibilités peut devenir très grand (en fait, ce nombre dépend directement du nombre de paramètres pris en compte par les méthodes). Dans la première version de CIME, la gestion des étapes et des chaînes par les règles, fait qu'en cas d'échec, aucun mécanisme de reprise n'est possible : la chaîne est abandonnée. Il aurait été souhaitable de pouvoir remettre en cause la dernière étape de la chaîne ayant conduit à l'échec. Ce rôle que jouent les règles, impose un déroulement très procédural de la recherche, et fait qu'il n'y a pas toujours une parfaite adéquation entre les données et la stratégie de recherche. Lorsque l'on met en place une chaîne, on **suppose** donc que les choix effectués sont bons, mais s'ils ne l'étaient pas, il faudrait pouvoir revenir sur ces choix, et en effectuer d'autres. Il a donc été décidé de gérer les chaînes et les étapes pour leur mise en place et leur remise en cause, en tant qu'**hypothèses**. Une hypothèse dans CIME.2 concernera donc la *méthode* (de classification par exemple), et les *arguments* de cette méthode. Le problème à résoudre est donc la gestion des hypothèses : leur émission, l'évaluation de leur pertinence, et leur remise en cause.

I.2. Présentation de l'algorithme

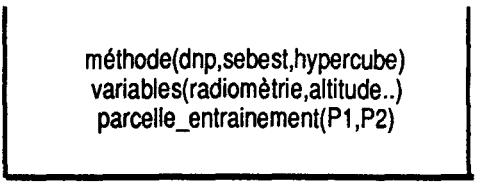
ADDB [De Kleer 86], pour Assumption-based Dependency-Directed Backtracking, est en fait un algorithme permettant de procéder à un ensemble de choix qui soit complet, permettant ainsi l'obtention d'une solution valide. Pour cela, ADDB gère une pile (i.e. : une structure dynamique de stockage d'informations) dont les éléments sont appelés **disjonctions de contrôle**, et qui représentent des ensembles d'assertions incompatibles deux à deux. Chacune de ces assertions constituera une **hypothèse du contexte**. L'utilisation de ADDB suppose que toute solution ne peut être valide que si elle se réalise dans un environnement (i.e. ensemble d'hypothèses) où exactement une assertion, issue de chaque disjonction de contrôle, a été prise comme hypothèse (c'est ce qui a été appelé ensemble complet de choix).

I.3. C.H. : Une adaptation de A.D.D.B. à CIME.2

Dans la nouvelle version de CIME, les hypothèses sont faites suivant un principe s'inspirant de ADDB. Rappelons que dans CIME.2 ce sont les

méthodes et leurs arguments qui constituent les hypothèses.

De ADDB, il a été retenu la gestion des disjonctions de contrôle qui permet :
=> d'effectuer toutes les combinaisons d'éléments pouvant être sélectionnés;
=> une évolution dynamique de l'espace de recherche, par ajout ou retrait de disjonctions de contrôle;
=> des retours arrière, sur les choix effectués, qui soient contrôlés;
=> d'être sûr d'avoir une solution se réalisant pour un ensemble de choix complet.
exemple : une pile de CIME peut être :



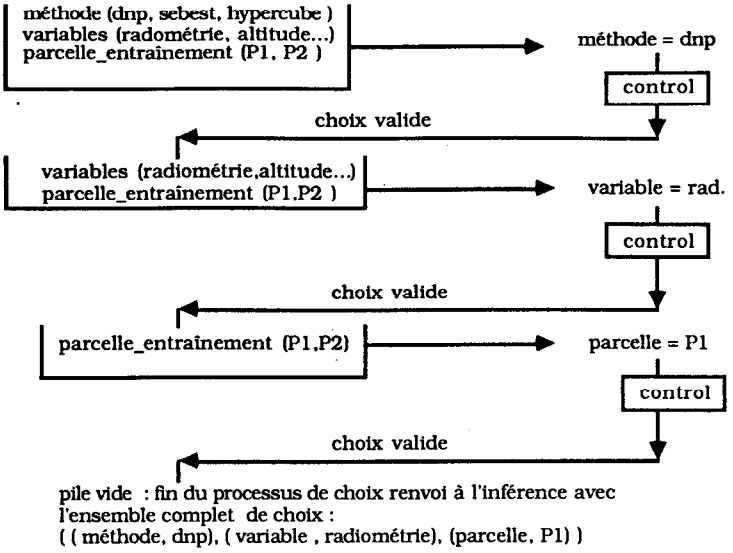
dans ce cas, un ensemble complet de choix peut être :
(dnp,altitude,P2)

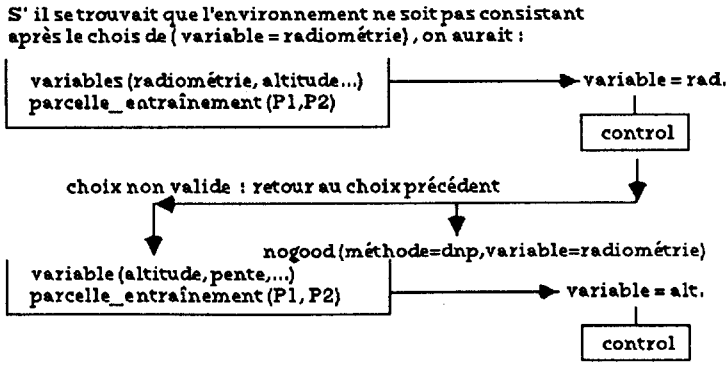
ou
(sebest,radiomètrie,P1)

I.4. L'algorithme

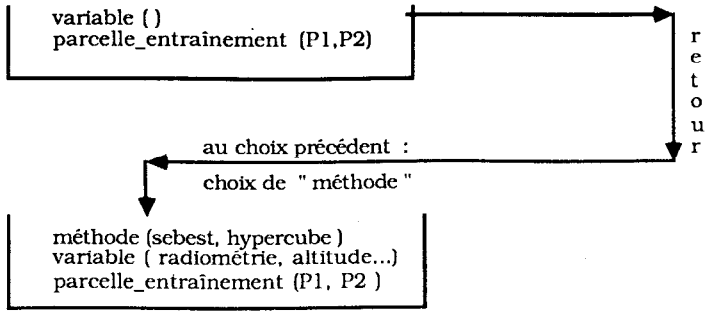
Le principe de l'algorithme est le suivant :
a) Partir d'un ensemble d'hypothèses initialement vide,
b) Si la pile est vide, l'environnement est complet =>le tester et éventuellement inférer (c'est à dire utiliser l'ensemble complet de choix obtenu pour l'étape)
c) Si la pile n'est pas vide, dépiler la disjonction en sommet de pile
d) Si la disjonction n'a plus d'élément => retourner au dernier choix effectué
e) Si la disjonction a encore des éléments => prendre le premier de ces éléments, et l'ajouter à l'environnement courant
f) Si ce nouvel environnement est consistant (i.e. il n'engendre aucune contradiction), recommencer en b)
g) Sinon, enregistrer l'environnement courant comme étant mauvais ("nogood") afin de l'éviter par la suite, et retourner au dernier choix effectué.

exemple de fonctionnement: en reprenant la pile précédante





dans le cas où la disjonction de contrôle "variable" ne contiendrait plus d'élément, il se passe la chose suivante :



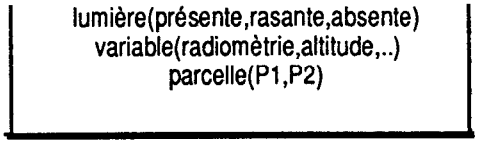
Plus aucun choix n'est possible lorsque l'on parvient "en retour" à la première disjonction et que celle-ci est vide : on a alors examiné toutes les possibilités.

Une particularité de cet algorithme, est la gestion dynamique des disjonctions de contrôle : en cours de session, de nouvelles disjonctions peuvent être ajoutées à la pile (on dit alors qu'elles deviennent *actives*), ou enlevées de la pile (on dit alors qu'elles deviennent *passives*). Cette gestion se fait à l'aide de règles d'inférences dites de contrôle, qui sont gérées par le moteur d'inférences comme les autres règles du système.

exemple : avec les mêmes éléments que précédemment, supposons que le choix de dnp implique la nécessité de prendre en compte l'ensoleillement (lumière:présente,rasante,absente). Dans ce cas on écrira la règle suivante :

si méthode = dnp
alors active(lumière(présente,rasante,absente))

Si effectivement dnp est choisie comme méthode, la pile devient :

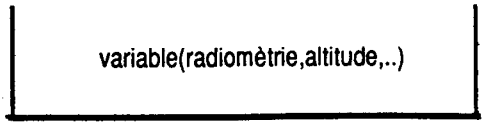


La disjonction "méthode" n'apparait pas ici puisqu'elle a été dépilée pour procéder au choix de (méthode = dnp). Si la méthode choisie n'était pas dnp, cette règle n'aurait pas été appliquée.

Inversement, si la méthode "sebest" ne nécessite pas le choix d'une parcelle d'entraînement, on écrira la règle suivante :

si méthode = sebest
alors passive(parcelle_entrainement)

On remarque qu'ici les valeurs d'assertions P1 et P2 ne sont pas utiles car c'est l'ensemble de la disjonction qui est inhibé. Dans le cas où "sebest" est sélectionnée, et la règles appliquée, la pile devient :



I.5. Les raisons du choix d'ADDB

Les raisons justifiant le choix de ADDB sont les suivantes :

- Sa *souplesse* qui permet de :
  1) rechercher une ou plusieurs solutions à un problème, sans qu'il y ait perte d'efficacité, comparativement aux autres méthodes. Dans la cas de CIME, ceci est appréciable car les différents sous-buts peuvent nécessiter des stratégies de recherche très différentes : on cherchera **toutes** les solutions quand il s'agira de classer les pixels de l'image pour produire une carte (on ne retient alors que la meilleure), ou **une** solution quand il s'agira d'en extraire des éléments caractéristiques, ou de focaliser la recherche.
  2) modifier dynamiquement la forme de l'espace de recherche, par ajouts

ou retraits de disjonctions de contrôle. Ainsi dans CIME.2, pourra-t'-on envisager de réduire l'espace de recherche au vu de certains résultats; faire appel à des procédures externes n'ayant pas toutes le même type, ni le même nombre d'arguments, sans avoir de gestion particulière à chaque cas.

- Ses *performances*
  Dûes au retour arrière chronologique qui permet de revenir sur le dernier choix effectué et non sur un choix quelconque. Ceci évite les retraits "aveugles", et des tentatives inutiles.

Il peut donc être intéressant, dans un système tel que CIME où le nombre de données traitées est grand et les traitements coûteux, d'expérimenter ces possibilités d'optimisations, et de mettre à profit les facilités offertes par cet algorithme.

II. T.M.S.

II.1. Les mises à jour dans la base de connaissances

A chaque nouvelle étape, les données utilisées dans le système sont réinitialisées, car leur grand nombre interdit toute duplication. Par ailleurs, au cours de l'étape, l'application des segments nécessite, pour tous les pixels, une nouvelle instanciation de la valeur de l'attribut "classe". Le processus classique de mise à jour consiste à enlever de la base de connaissances les faits correspondant aux anciennes valeurs, et à insérer ceux correspondant aux nouvelles, afin de toujours conserver une base de connaissances cohérente.

Il est important de remarquer que le principe de fonctionnement impose la nécessité de toujours avoir une base de connaissances **cohérente.** Cela signifie qu'aucune contradiction ne doit exister dans les faits contenus dans la base (par exemple l'existence simultanée des faits X=1 et X=2, constitue une contradiction). La conséquence d'une contradiction dans un système déductif tel que CIME, c'est que des inférences pourraient avoir lieu sur des données erronées (ie. participant à la contradiction), ce qui entraînerait la production d'autres faits eux même erronés.

Dans CIME.1, la procédure chargée de ces mises à jour, procède au retrait de tous les faits de la base de connaissances, et en se basant sur les dates de validation, par la suite, valide de nouveau les faits ne devant pas être remis en cause. Etant donné le nombre important de faits devant à chaque fois être "revalidés" et la fréquence de ces mises à jour, le temps d'exécution de cette procédure est particulièrement important.

II.2. Présentation de la structure utilisée

Afin d'optimiser le processus de mise a jour, des informations précises sont nécessaires pour un fait donné F :

* Les faits permettant de l'obtenir
* Les faits qu'il permet de déduire

et parmi ces deux ensembles il faut repérer les éléments nécessaires et ceux

qui ne sont que suffisants.

exemple :

Base de faits initiale (A,B,D,E)

Base de règles = R1 : A,B -> C (si A et B alors C)
R2 : D -> C
R3 : B,E -> F
R4 : C -> G,H
R5 : F -> G
R6 : B -> E

après application de ces règles la nouvelle base de faits est :
(A,B,C,D,E,F,G,H)

si l'on est amené à enlever le fait B de cette base, cela entraînerait le seul retrait de F puisque C peut toujours être déduit par la règle R2.

T.M.S. [Dougle 79], pour Truth-Maintenance System, utilise une structure de données particulière lui permettant d'avoir accès, pour chaque fait F, à toutes ces sortes d'informations qui mettent en évidence les liens de dépendance entre les données (data dependencies), et qui est la suivante :

(F,Liste des justifications,Liste des justifiés,Liste des supporteurs,Liste des supportés,Statut)

où :

- **Liste des justifications** : liste des faits ayant permis par leur présence (IN-justifications), ou par leur absence (OUT-justification), de dériver F.
- **Liste des justifies** : Liste des faits ayant F dans leurs justifications.
- **Liste des supporteurs** : Liste des faits issus de la liste des justifications, et choisis comme support principal.
- **Liste des supportes** : Liste des faits ayant F dans leur liste des supporteurs.
- **Statut** : a pour valeur **IN** ou **OUT**, suivant que le fait est ou non correctement justifié.

Avec cette structure, les faits initiaux (ou prémisses) ont une liste des justifications vide. Pour qu'un fait F ait un statut à IN, il faut qu'il ait au moins une chaîne de justifications non circulaire, partant de F, et aboutissant à un fait initial ou à une hypothèse. D'où la notion de WFS, pour Well Founded Support, dont le supporteur est un représentant. Pour un fait de statut IN, plusieurs chaînes de justifications peuvent exister. Lorsque le statut d'un fait est à OUT, la liste des supporteurs contient un représentant de chaque élément de la liste des justifications.

Les contradictions sont détectées par le module déductif, qui en informe TMS en lui fournissant la liste des faits (justifications) engendrant la contradiction.

exemple : avec les bases initiales précédentes on a:

(A,[],[C],[],[],IN)

(B,[],[C,F],[],[],IN)

II.3. Présentation de l'algorithme

Les mises à jour, gérées par TMS, s'effectuent lorsque le statut d'un fait passe de **IN** à **OUT** par invalidation d'une assertion; ou, inversement, lorsque ce statut passe de OUT à IN par ajout de nouvelles justifications.

Algorithme de gestion des ajouts et des retraits de faits : D.B.G.C. (Data Base Garbage Collector)

Lors de ces mises à jour, les faits supportés sont examinés, leur liste de supporteurs et leur statut recalculés, et ceci récursivement. Les faits ajoutés sont transmis par le module déductif avec leurs justifications, les faits à enlever sont simplement signalés à TMS, et la mise à jour se fait comme suit :

* Ajout de F et de <Liste de justifications>

Il y a mise à jour des justifiés de F, par introduction de F dans la liste des justifications.

exemple : application de R1 => ajout de C + [(A,B)]

d'où : (C,[(A,B)],[G,H],[A,B],[],IN)

(A,[],[C],[],[C],IN)

(B,[],[C],[],[C],IN)

mise à jour de G et H :

(G,[C],[],[],[],OUT)

(H,[C],[],[],[],OUT)

* Suppression de F

exemple : suppression de (B,[],[C,E,F],[],[C,F],IN)

Pour chacun des justifiés F'de F (ici pour C,E et F)

->*Si F' n'est pas supporté par F* :il y a mise à jour de ses justifications, par suppression de toutes celles contenant F.

exemple : c'est le cas de E qui devient :

(E,[],[F],[],[],IN)

E est un fait initial, et sauf retrait explicite, il demeure valide.

->*Si F' est supporté par F* (cas de C et F)

-Le statut de F' passe de **OUT** à **0** afin de rechercher un autre supporteur pour F' (Le statut **0** est un statut **OUT** provisoire, qui ne sert que le temps de mise à jour).

-F' est mis provisoirement dans une liste L de faits à étudier.

-Ce même processus est appliqué récursivement aux supportés de F'.

Remarque : L'utilisation du statut provisoire 0 et de la liste d'examen L, sert à éviter les justifications circulaires.

exemple : F -> F'

F' -> G

G -> F'

Ce cycle s'arrête de deux façons :

1°) Un élément de la liste L a son statut qui passe de **0** à **IN** : un autre supporteur lui a été trouvé. Dans ce cas, tous les éléments de liste sont réexaminés.

2°) On est arrivé à des faits initiaux ou à des hypothèses sans qu'aucun élément de la liste ait eu son statut changé. Alors, F' est affecté d'un statut **OUT**, ainsi que tous ceux de la liste.

fin de l'exemple :

* Premier pas : les statut de C et F passent provisoirement à 0 et L = (C,F)
* Examen de C : un autre supporteur lui est trouvé : c'est D
* => Le statut de C passe de 0 a IN
* Réexamen de la liste L = (F)
* Réitération du processus => L = (F,G) (car G est supporté par F)
* Un autre supporteur est trouvé pour G : c'est C
* => Le statut de G passe de 0 a IN
* Réexamen de la liste L = (F)

=> Plus aucune itération n'est possible => F est doté d'un statut OUT définitif.

BIBLIOGRAPHIE

MERING C., BLAMONT D., GANASCIA J.G., MONJANEL F. : CIME : Une application des systèmes experts à la télédétection. Actes des 8ème rencontres internationnales d'Avignon "Les systèmes experts et leurs applications"; 1988.[Blamont 88]

DE KLEER J. and B.C. WILLIAMS : Back to Backtracking : Controlling the ATMS. Actes du colloque AAAI, vol 2,pp 132-139;1986.[de Kleer86]

DOYLE J. : A Truth-Maintenance System. Artificial Intelligence, vol 12,3;1979.[Doyle79]

ETIFIER A. : rapport de D.E.S.S. : Application des systèmes experts à la cartographie par télédétection, Université d'Orsay L.R.I., 1988.[Etifier88]

MONJANEL F. : rapport de D.E.S.S : Système expert appliqué à la télédétection, Université d'Orsay L.R.I., 1987.[Monjanel87]

ORSTOM Éditeur
Dépôt légal : janvier 1990
Impression - ORSTOM BONDY