LABORATOIRE D'INFORMATIQUE THEORIQUE
& APPLICATIONS DE MARSEILLE
L.I.T.A.M.
Faculté des sciences Economiques

UNIVERSITE d'AIX-MARSEILLE II

ISSN 0291 - 5413

INFORMATIQUE FONDAMENTALE & APPLICATIONS

BULLETIN 26

SOMMAIRE

P1 ... EDITORIAL:
Informatique et détournement.. E. Bianco

P4 ... Mathématiques du multicritère:
R. Cusin

P22 ...La machine à instruction unique:
E. Bianco

P30 ...Amélioration de l'algorithme de production.
E. Boudiba.

P46 Vouzzavedibisar: L'Horreur.
E. B.

JUIN 1990

Adresse postale : FACULTé DES SCIENCES ECONOMIQUES
LITAM
14 rue Puvis de CHAVANNES 13001 MARSEILLE
91 90 13 20 P 420 et 421

INFORMATIQUE ET DETOURNEMENT.

E. Bianco

Les civilisations monothéistes qui nous cernent sont génératrices de détournements en tous genres.

Une telle assertion ne devrait pas avoir à être illustrée, mais que voulez-vous, de même que le poisson est le seul animal à ignorer la soif, le mammifère aérien ne possède pas de mot courant pour désigner un manque d'air passager. A force de baigner dans un fluide abominablement délétère, on finit pas ne plus percevoir l'odeur, même si l'on doit en crever ... sinon depuis belle lurette plus personne n'habiterait les villes.

On est immergé depuis dès avant l'enfance dans un océan de détournement. On naît d'un détournement, on est nourri spirituellement et matériellement au biberon du détournement, et plus tard, blanchi sous le harnois, on vit journellement du pain du détournement et pour survivre au delà du principe de Peters, on détourne quelque peu les classes montantes.

Malheur à ceux qui n'ont pas compris le jeu ou à qui ce jeu déplaît.

Cela commence bien avant la conception. Là où les animaux pratiquent la pariade, les humains pratiquent le détournement par intoxication. Les oripaux criards qui masquent les déficiences physiques, les verbiages abondants qui masquent le vide de la pensée, la grosse voiture, le yacht ou le faux-vrai superbe appartement plastico-pseudo-bourgeois qui masquent tout le reste. Et cela suffit à justifier cette frénésie d'ascension sociale à grouillement de panier de crabes.

La deuxième phase est d'un double détournement, le premier qui consiste, après le dépassement de la première illusion, à tenter de s'attacher le conjoint qui se tire en douce par l'argument fort de la procréation _ détournement de l'amour _ le second qui favorise le premier à l'aide de "primes" pour maintenir un bon niveau de clientèle à l'économie de marché _ détournement de la procréation _ .

Le petit d'homme est là, maintenant. De plus en plus encombrant dans notre société qui se complique vertigineusement.

Tout lui est objet de tentation, en plus de ses pulsions naturelles dont on ne sait plus très bien ce qu'elles représentent dans un fouillis devenu inextricable. C'est devenu un cliché, un leit-motiv, un poncif, un lieu commun d'évoquer l'enfant devant la télévision. C'est moins courant d'observer les parents dans leur numéro acrobatique de glissement, de

fuite, de détournement de conversation. L'apprentissage commence. Si le jouet trop coûteux déclenche la crise, on glisse vers le bonbon qui fait pschitt dans la bouche, de l'excès de bonbons on passe à la belle dame qui raconte une bien belle histoire, quand la dame attire trop l'attention... Le tout sur fond de calottes de plus en plus remplacées par des menaces subtiles et dangereuses.

Devenu adolescent, l'enfant a pris le pli, mais ses goûts ont changé. il s'intéresse à l'autre sexe, cependant le réflexe de détournement est si fort que la fillette va répondre un NON violent aux avances du garçon qui lui plaît, quand celui-ci ose lui en faire. Le garçon reçoit une fille avec des manières d'autant plus rogues, que celle-ci l'attire davantage. Les situations d'attraction-répulsion peuvent se combiner à l'infini, en provoquant parfois des attitudes et des situations ahurissantes. A cela il faut ajouter que de détournement en détournement, garçons et filles chacun de leur côté se sont constitué une spécialité sociale qui en font des étrangers les uns par rapport aux autres, voire des espèces différentes.

Plus grand, les choses se décantent, et selon les natures, les caractères les frustrations causées par ces tamponnements de sentiments contradictoires, créent tout l'arc-en-ciel des personnalités entre le Don-Juan impénitent et perpétuellement insatisfait, et le bloqué complet retiré du monde dans sa paranoïa écorchée vive. Avec tous les laissés-pour-compte et toutes les ascensions sociales fracassantes, tous les micro-potentats dans leurs micro-citadelles imprenables et qui, soudain nus et vidés de leur pâle substance, s'étiolent et meurent dès la retraite.

Et salvatrice est apparue l'informatique.

Dans l'immense fracas de la solution définitive à la communication, le Dieu nouveau s'est révélé.

COMMUNICATION est un étrange mot dont la signification me semble de même nature que l'attitude que se donnent certains animaux pour paraître autre chose. Le caméléon la plie ou la limande se fondent dans le décor en adaptant leur couleur. Certains insectes ont des formes de feuilles ou bien de branchettes. Le mot soif présente également cette versatilité, la soif des réclames d'apéritifs est presqu'opposée dans son sens à la soif organique qui ne peut s'étancher qu'avec de l'eau. Autre détournement.

Le gigantesque tamtam à support d'informatique qui sillonne le globe en tous sens en réunissant des masses d'ordinateurs, me paraît peu adapté à satisfaire la soif de communication dont crèvent peu ou prou nombre de nos congénères. Mais cet immense bruit ne joue-t-il pas le même rôle que le tambour et le clairon sonnant la charge et certaine musique qui marche au pas-de-l'oie, et le petit coup de gnôle mêlée

d'éther du brave fantassin de quatorze montant à l'assaut ? Détournement, toujours détournement.

Alors que voila une moyen formidable de remettre à leur juste niveau les matamores indélicats dont la seule préoccupation est de nous croire et de tenter de nous maintenir à leur niveau de débilité. Il suffirait pourtant de savoir dire non et de le faire savoir par delà les frontières et les océans pour que tous ces chiffons qui ne rêvent que de statues équestres dans les carrefours pour laisser une trace éternelle de leur pauvre existence de forcenés qui ont oublié de vivre, soient ramenés à leur véritable valeur, celle que leur reconnaissent les pigeons.

Le bipède frileux qui a pu accumuler mille raisons de ne plus vouloir s'adresser à ses semblables, a enfin découvert l'interlocuteur agréable et attentionné dont il n'osait plus rêver. Une masse importante d'humains de tous acabits, nous dit-on, s'est refermée sur son ordinateur, qui travaillent nuit et jour, ne sortant plus, ne parlant plus à personne, sinon à leur chère machine, ne vivant plus que pour leur ordinateur. Après la reconnaissance des mariages homosexuels il est évident que si cette humanité n'était pas si coupée du monde elle ne tarderait pas à exiger la reconnaissance du mariage avec un ordinateur ou une ordinatrice.

Pour qui n'a jamais vu à la télévision une tête de premier ministre de Président, voire de ministre de l'éducation nationale, ou encore pour qui n'a jamais vu interville publicité ou la caméra invisible, tout cela pourrait paraître bien étrange. Amis nous vivons les riches heures d'une bien riche époque.

Introduction aux **MATHEMATIQUES DU MULTICRITERE**

Les modèles fondés sur l'optimisation et servant à la prise de décision, se sont complexifiés depuis une vingtaine d'années environ. Avant les années 70, de tels modèles se ramenaient souvent à optimiser une fonction définie sur une partie X de $\mathbf{R}^n$ et à valeurs dans **R** ; Une simplification supplémentaire consistait à supposer f linéaire et X polyèdre (c'est l'étude d'un problème de programmation linéaire). Les modélisateurs se sont rendus compte de l'imperfection très grande des modèles ainsi élaborés: une des raisons essentielles en était la réduction à un seul critère sur lequel se fondait la décision. Comment concilier l'idée de vouloir, par exemple, maximiser l'investissement, avec l'idée de maximiser le profit et de minimiser le chômage, ... , en s'appuyant sur un seul critère à optimiser ? A l'évidence, la prise en compte simultanée de plusieurs critères devenait nécessaire. Ainsi est née ce que l'on peut appeler "l'optimisation multicritère"; Bien que cette appellation fort ambiguë ne fasse pas l'unanimité (surtout auprès des praticiens) !

En fait, face à des critères plus ou moins conflictuels, il s'agit de proposer une ou plusieurs solutions à un décideur; ces solutions étant les meilleures en un sens à définir.

L'article qui suit se propose de présenter certaines bases mathématiques, utilisées dans "l'optimisation multicritère". Il est extrait d'un ouvrage en cours de rédaction, s'appuyant sur un enseignement de DEA délivré au GREQE à Marseille.

A cause d'incidents typographiques intervenus dans notre dernier numéro, et dont nous nous excusons auprès de l'auteur, nous préférons reprendre la première partie déja imprimée dans le numéro 25 et publier dans son entier l'article de Roger Cusin dans ce présent numéro.

Les mathématiques du multicritère

2.1. Préordre et ordre sur un ensemble

Soit X un ensemble non vide et soit R une relation binaire sur X. R est appelée un **préordre**, si les 2 axiomes suivants sont vérifiés:

(0_1) $\forall x \in X$, xRx est vraie (réflexivité);

(0_2) $\forall (x,y,z) \in X^3$, (xRy et yRz) = (xRz) (transitivité)

Le préordre est dit **total** si, en outre:

$$\forall (x,y) \in X^2, \text{ xRy ou yRx.}$$

Un ensemble muni d'un préordre est dit préordonné; et s'écrit $(X, \underline{\alpha})$. Nous noterons $x \underline{\alpha} y$ pour xRy.

Dans le cas particulier où l'axiome (0_3) est vérifié, le préordre est appelé un **ordre**:

(0_3) $\forall (x,y) \in X^2$, $(x \underline{\alpha} y$ et $y \underline{\alpha} x) \Rightarrow x=y$ (antisymétrie)

Au symbole "$\underline{\alpha}$" du préordre on substitue, dans le cas de l'ordre, le symbole "$\leq$", et l'on écrit: $x \leq y$.

Un ensemble muni d'un ordre est dit ordonné; et s'écrit: $(X, \leq)$.

Rappelons que l'axiome (0_3) est indépendant des 2 autres; ce qui signifie qu'il existe des ensembles préordonnés pour lesquels le préordre n'est pas un ordre. Par exemple, $\mathbb{Z}$ muni de la relation "divise" est un ensemble préordonné pour lequel le préordre "divise" n'est pas un ordre puisque 1 divise -1 et -1 divise 1.

Donnons un second exemple important pour l'optimisation multicritère:

• Soit X un ensemble non vide et $f: X \to \mathbb{R}$. Définissons sur X la relation binaire $\underline{\alpha}$ par:

$$x \underline{\alpha} y \Leftrightarrow f(x) \leq f(y).$$

- $\underline{\alpha}$ est un préordre total sur X;
- $\underline{\alpha}$ n'est pas un ordre car:

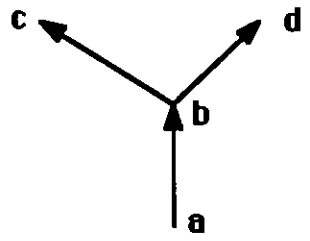
Si $x \underline{\alpha} y$ et $y \underline{\alpha} x$, alors f(x) = f(y); cela n'implique pas que x soit égal à y puisque f est quelconque (non injective).

• Soit (X,α) un ensemble préordonné, x **domine** y, si $y\alpha x$. Dans le cas d'un ordre $\leq$, on dit que x est **supérieur ou égal** à y. Si x est **supérieur ou égal** à tout $y \in X$, x est appelé **plus grand élément de X**; il est unique à vérifier cette propriété. Les notions d'**inférieur ou égal** et de **plus petit élement** se définissent naturellement en inversant les rôles de x et de y.

L'ordre strict associé à "$\leq$" se note "$<$". Cette notion de plus grand élément est malheureusement trop restrictive: peu d'ordres intéressants en pratique, possèdent un plus grand élément. Par contre, nombreux sont les ordres avec éléments maximaux. On dit que, pour la structure préordonnée (X,α), x est un élément **maximal,** s'il n'existe pas $y \in X$ tel que: $x\alpha y$ et $°(y\alpha x)$. Autrement dit, x est maximal si, dès lors que $x\alpha y$, on a aussi $y\alpha x$. La définition vaut également pour un ordre $\leq$. Dans ce cas , dire que x est **maximal** équivaut à dire qu'il n'existe pas y tel que $x \leq y$ et $y \neq x$. Un élément y de X est donc soit incomparable avec x, soit inférieur ou égal à x. Une autre façon d'exprimer que x est maximal est de dire qu'il n'existe pas y de X strictement supérieur à x.

2.2. Etude de la maximalité pour les ensembles ordonnés

S'il existe un plus grand élément x de X, il est clair que c'est un élément maximal et que c'est le seul. Par contre, des éléments maximaux peuvent exister dans le cas où il n'existe pas de plus grand élément. Pour s'en persuader, faisons appel au **diagramme de Hasse de la relation d'ordre:**



Un tel diagramme doit être lu de la façon suivante:

$a < b < c$, $b < d$, c et d incomparables.

Dans cet exemple, c et d sont 2 éléments maximaux. Il est clair qu'un ensemble ordonné **fini** X possède au moins un élément maximal. Il suffit de partir d'un élément x_0 de X et de raisonner ainsi: si x_0 n'est pas maximal, il existe x_1, $x_0 < x_1$: si x_1 n'est pas maximal, on recommence l'opération précédente. Comme X ne contient qu'un nombre fini d'éléments, le processus s'arrête nécessairement sur un élément maximal.

Lorsque X est infini, la recherche des éléments maximaux est d'une grande complexité. Le théorème de Zorn donne une condition suffisante d'existence d'éléments maximaux.

Soit A une partie non vide de X; x est un **majorant** de A, si pour tout a de A, $a \leq x$. Dans le cas où x est le plus petit majorant de A, x est appelé la **borne supérieure** de A.

L'ensemble ordonné X est dit **inductif**, si toute partie totalement ordonnée de X (pour l'ordre induit sur la partie par l'ordre initial $\leq$ sur X) possède une borne supérieure.

Théorème de Zorn. Tout ensemble ordonné inductif possède un élément maximal.

La démonstration de ce théorème fondamental repose sur le lemme suivant:

Lemme: Soit E un ensemble ordonné et f une application de E dans E telle que $f(x) \geq x$ pour tout $x \in E$. Soit $\mathfrak{F}$ l'ensemble des parties X de E telles que:
1°- $f(X) \supset X$,
2°- Si $Y \supset X$ possède une borne supérieure dans E, cette borne supérieure appartient à X.

Sous ces hypothèse, si $a \in E$, l'intersection C_a des $X \in \mathfrak{F}$ contenant a appartient à $\mathfrak{F}$ et, si $x \in C_a$ et $y \in C_a$, on a $y \leq x$ ou $y \geq f(x)$; en particulier C_a est totalement ordonné.

Preuve du Lemme:

a) Il est clair que C_a appartient à $\mathfrak{F}$ et, comme l'ensemble des $x \geq a$ appartient à $\mathfrak{F}$, a est donc le plus petit élément de C_a.
b) Soit D l'ensemble des $x \in C_a$ tels que: $(y \in C_a$ et $y \leq x) \Rightarrow (y = x$ ou $f(y) \leq x)$.

Pour $x \in D$, soit C'x l'ensemble des $y \in C_a$ tels que $y \leq x$ ou $y \geq f(x)$. Il est clair que $a \in C'_x$ si $x \in D$. On va montrer que, pour tout $x \in D$, $C_a = C'_x = D$. Pour cela, il suffit de montrer que $C'_x \in \mathfrak{F}$ et que $D \in \mathfrak{F}$.
c) Soit $x \in D$. C'_x vérifie 1°- En effet, soit $y \in C'_x$; si $y \geq f(x)$, on a $f(y) \geq y \geq f(x)$ et, si $y \leq x$, on a $x=y$ ou $f(y) \leq x$ (car $x \in D$); dans tous les cas, on a $f(y) \in C'_x$.

C'_x vérifie 2°- Soit Y une partie de C'_x ayant une borne supérieure b dans E; $b \in C_a$. Si, pour tout $y \in Y$, on a $y \leq x$, alors $b \leq x$. Dans le cas contraire, il existerait $y \in Y$ tel que $y \geq f(x)$ et $f(x) \leq b$; ainsi, $b \in C'_x$. Il en résulte alors que $C'_x = C_a$.

d) D vérifie 1°- Si $x \in D$, soit $y \in C_a$ tel que $y \leq f(x)$, comme $y \in C'_x = C_a$ (d'après c), on a $y \leq x$ ou $y \geq f(x)$.
Dans le premier cas, $y=x$ et donc $f(y) = f(x)$, ou $f(y) \leq x \leq f(x)$ et, dans le second cas, on a $y=f(x)$; par conséquent $f(x) \in D$.

D vérifie 2°- Soit Y une partie de D ayant une borne supérieure b dans E; pour tout $x \in Y$, on a $f(x) \leq b$. En effet, comme $x \in D$ et que $b \in C_a = C'_x$, on a $b \geq f(x)$, car sinon $b \leq x$ ce qui conduit à b=x, pour tout x: absurde. Soit alors $y \in C_a$ tel que $y \leq b$; si, pour tout $x \in Y$, on a $f(x) \leq y$, alors $x \leq y$, et donc y=b. S'il existe $x \in Y$ tel que l'on n'ait pas $f(x) \leq y$, alors $y \leq x$ puisque $y \in C'_x$. Comme $y \in C_a$, $y = x \leq b$ et donc $f(y) = f(x) \leq b$. L'autre éventualité est que $f(y) \leq x$; d'où: $f(y) \leq x \leq f(x) \leq b$. Par conséquent $b \in D$.

Corollaire 1 (Théorème du point fixe)

Soit E un ensemble inductif et f une application de E dans E telle que $f(x) \geq x$ pour tout $x \in E$. Alors, il existe un élément b de E tel que $f(b) = b$.

Preuve du Corollaire

Soit $a \in E$; comme E est inductif et que, d'après le lemme, C_a est totalement ordonné, C_a possède une borne supérieure b dans E qui appartient à C_a car $C_a \in \mathfrak{F}$. On a ainsi $f(b) \in C_a$ et donc $f(b) \leq b$. Mais comme par ailleurs, $f(b) \geq b$; $f(b) = b$.

Démonstration du Théorème de Zorn

Soit E un ensemble inductif et $x \in E$. Si x est maximal, posons f(x)=x, si x n'est pas maximal, soit f(x) un élément y de E tel que $f(x) > x$. On définit ainsi une application f de E dans E telle que $f(x) \leq x$ pour tout $x \in E$. Le théorème est alors une conséquence du corollaire 1 précédent, puisque la relation "f(x)=x" est équivalente à "x est maximal".

Corollaire 2

Soit E un ensemble inductif et $a \in E$; alors, il existe un émément maximal x de E tel que $x \geq a$. Il suffit de remarquer que l'ensemble des éléments y de E tels que $y \geq a$, est inductif.
Remarque: Une notion d'ensemble inductif en apparence plus faible que la précédente, peut être définie; elle conduit au même résultat. De façon précise, nous dirons que E est **faiblement inductif,** si toute partie non vide totalement ordonnée admet au moins un majorant. Le théorème de Zorn reste encore valable pour les ensembles faiblement inductifs:

Corollaire 3

Si E (non vide) est faiblement inductif; alors E possède un élément maximal.

Preuve

Soit A l'ensemble de tous les sous-ensembles non vides et totalement ordonnées de E. A n'st pas vide car tout singleton est élément de A. Si X et Y$\in$ A, définissons X$\leq$Y par la condition X$\subset$ Y. Il est clair que A est ordonné. Montrons que A est inductif. Soit $(X_i)_{i\in I}$ un sous-ensemble totalement ordonné de A et posons $Z = \bigcup_{i\in I} X_i$. Z est totalement ordonné car

si x et y$\in$Z, il existe i$\in$I et j$\in$I tels que x$\in X_i$ et y$\in X_j$. $(X_i)_{i\in I}$ étant totalement ordonné, on a, par exemple $X_i \subset X_j$. Il s'ensuit que x et y$\in X_j$ et comme X_j est totalement ordonné, que x$\leq$y ou que y$\leq$x. Par conséquent, Z est totalement ordonné et c'est évidemment la borne supérieure de $(X_i)_{i\in I}$ dans A. D'après le théorème de Zorn, A possède un élément maximal X_o. Cela signifie que X_o est un sous-ensemble de E totalement ordonné et maximal pour l'inclusion. Soit m un majorant de X_o; montrons que m est maximal dans E. Soit en Effet, x$\in$ E, tel que m$\leq$x. $X_o \cup \{x\}$ est totalement ordonné et donc egal à X_o puisque X_o est maximal dans A. Par conséquent, x$\in X_o$ et x$\leq$m. Donc x=m; ce qui prouve que m est maximal.

2.3. Ordre associé à un préordre

Considérons un préordre $(x,\underline{\alpha})$; la relation binaire S défini par:

$$xSy \Leftrightarrow x\underline{\alpha}y \text{ et } y\underline{\alpha}x$$

est une relation d'équivalence sur X.

Sur l'ensemble quotient X/S formé des classes d'équivalences x, y, ... définissons la relation binaire $\leq$ par:

$$\bar{x} \leq \bar{y} \Leftrightarrow \exists\, x\in\bar{x} \text{ et } \exists\, y\in\bar{y},\ x\underline{\alpha}y.$$

Il est prouvé, aisément, que $\leq$ est un ordre sur X/S; celui-ci est appelé **ordre canonique associé au préordre $\underline{\alpha}$.**

Il résulte de la définition posée que x est maximal pour le préordre α si et seulement si x est maximal pour l'ordre canonique associé. Cette remarque permet de restreindre l'étude de la maximalité sur un ensemble préordonné $(X,\underline{\alpha})$ à l'étude de la maximalité sur un ensemble ordonné; c'est ce que nous ferons dans le paragraphe suivant.

Auparavant, mentionnons un théorème dont les hypothèses sont souvent vérifiées en pratique. Il permet d'affirmer l'existence d'éléments maximaux.

Théorème 1 Soit X un espace topologique compact muni du préordre $\underline{\alpha}$. Supposons que pour tout $x \in X$, l'ensemble $F_x = \{y / x \underline{\alpha} y\}$ soit fermé; alors, pour tout $x \in X$, il esiste un élément maximal x_o tel que $x \underline{\alpha} x_o$.

Preuve

Il suffit pour cela de montrer que le préordre $\underline{\alpha}$ est faiblement inductif. Soit $(X_i)_{i \in I}$ une partie totalement préordonnée de X; F_{xi} est fermé par hypothèse, quel que soit $i \in I$. Il en résulte que $\bigcup_{i \in I} F_{xi} = F_I$ est un ensemble fermé de X, et donc compact.

Supposons que $F_I = \emptyset$; comme X est compact, il existe un nombre fini de F_{x_i} dont l'intersection est vide:

$$F_{x_1} \cup ... \cup F_{x_k} = \emptyset.$$

Supposons, puisque $(X_i)_{i \in I}$ est totalement préordonnée, que: $x_1 \underline{\alpha} x_2 \underline{\alpha} \ ... \ \underline{\alpha} x_k$; cela signifie que $F_{xk} \subset F_{x1}$, et donc que $F_{xk} = \emptyset$. Cette propriété est absurde puisque $x_k \in F_{xk}$. Par conséquent, $F_I \neq \emptyset$. Si $y \in F_I$, y est un majorant des $(X_i) i \in I$. Le théorème de Zorn permet de conclure.

2.4. Etude d'une structure particulière, fondamentale en optimisation multicritère

Soit X un ensemble non vide et soient p fonctions $f_1, ..., f_p$ définies sur X et à valeurs réelles:

$$f_i : X \to \mathbb{R}, \forall \ i=1, ..., p.$$

Munissons X de la relation binaire R définie par:

$$xRy \Leftrightarrow f_i(x) \leq f_i(y) \quad \forall \ i=1, ..., p.$$

y est dit "unanimement préféré à x".

Il est aisé de vérifier que R est une relationde préordre que nous noterons désormais $\underline{\alpha}$. Nous dirons que $x \in X$ est un **optimum de Pareto** pour $\underline{\alpha}$, si x est maximal. Un tel élément x est encore nommé **efficace** ou **non-dominé.**

Dans la suite nous privilégierons le terme efficace. L'efficacité d'un élément x de X se caractérise par le fait qu'il n'existe pas y de x tel que $f_i(y) \geq f_i(x)$ pour tout i, i variant de 1 à p,

l'une au moins des inégalités précédentes étant stricte. En d'autres termes, si x est efficace et si $f_i(x) < f_i(y)$ pour un i, il existe j tel que $f_j(x) > f_j(y)$.

Remarques

1) Dans la construction précédente de la structure préordonnée $(X, \underline{\alpha})$, la finitude de l'ensemble des fonctions (nous avons supposé qu'il y en avait p) n'intervient pas. Nous aurions donc pu procéder avec un ensemble d'indices I non nécessairement fini; en pratique, toutefois, nous n'aurons à faire qu'à des ensembles finis d'indices.

2) Aucune hypothèse n'ayant été faite sur les fonctions f_j (injectivité, surjectivité, ...) un élément efficace x n'est pas, en général, un plus grand élément et le préordre n'est pas un ordre. Dans le cas d'une seule fonction f(p=1), si f est injective, le préordre α est un ordre et un élément efficace est le plus grand élément de X. Il n'en existe, dans ce cas, qu'un au plus.

2.5. Optimum de Pareto-faible

Soient X un ensemble non vide et p fonctions réelles définies sur X; comme cela a déjà été envisagé en (2.3).

Introduisons sur X une nouvelle relation binaire P définie par:

$$xPy \Leftrightarrow f_i(x) < f_i(y) \quad \forall\, i=1, \ldots, p.$$

P n'est plus un préordre. Nous poserons toutefois la définition suivante:

$x \in X$ est un **optimum de Pareto-faible** s'il n'existe pas y tel que "xPy" soit soit vraie. Si nous notons alors P l'ensemble des optima de Pareto de X et P_f l'ensemble des optima de Pareto-faible; il est clair que $P \subset P_f$. Il existe des contre-exemples montrant que ces 2 notions ne sont pas équivalentes dans le cas général, nous en donnerons un ci-dessous.

2.6. Représentations géométriques dans l'espace de départ

Nous nous placerons dans la situation où X est une partie de $\mathbb{R}^2$ et où p=2. Nous pourrons ainsi avoir une visualisation des concepts et des résultats proposés.

1°- A titre d'exemple, montrons qu'un optimum de Pareto-faible n'est pas nécessairement un optimum de Pareto. Pour cela, considérons le domaine X représenté dans la figure 1; les 2 fonctions objectifs sont de la forme $z_1 = \langle c_1, x \rangle$ et $z_2 = \langle c_2, x \rangle$ avec c_1 "porté" par l'axe ox_1 et c_2 "porté" par l'axe ox_2. Il est clair que le point x° est un exemple d'optimum de Pareto-faible qui n'est pas un optimum de Pareto puisqu'en x_1 on améliore strictement z_1 sans détériorer z_2.

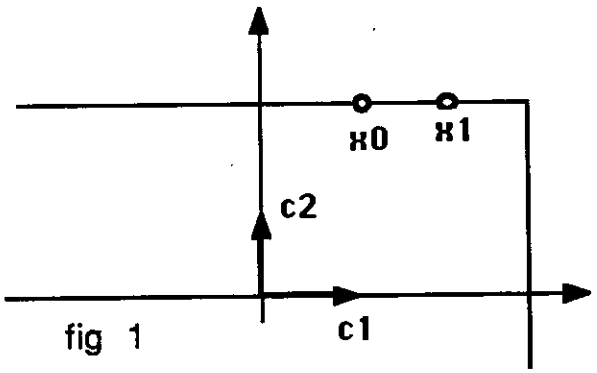


fig 1

2°- Dans l'exemple de la figure 2, les optima de Pareto sont situés sur les morceaux de frontières du domaine, AB et CD.

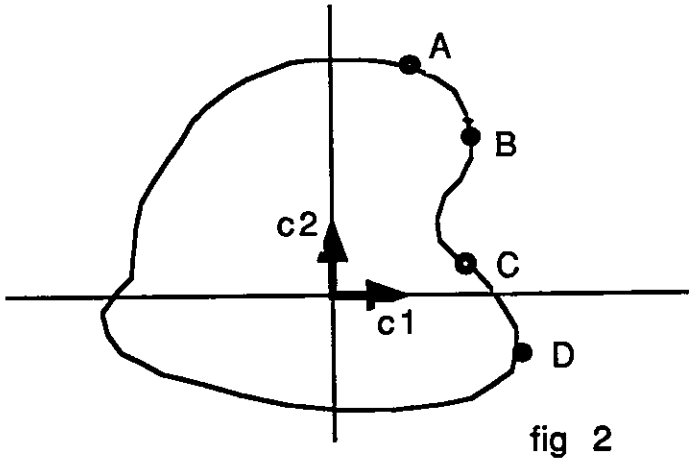


fig 2

Ce simple exemple montre la complexité de l'ensemble des optima de Pareto. Il est en effet exclu que l'on puisse en obtenir une caractérisation au moyen d'ensembles possédant de "bonnes propriétés", comme par exemple les ensembles convexes. Toutefois, dans le cas linéaire, des propriétés particulières permettront l'identification de cet ensemble.

2.7. Propriété fondamentale des optima de Pareto. Théorème fondamental

1) S'il existe des scalaires $\lambda_1 \geq 0, ..., \lambda_p \geq 0$ non tous nuls et tels que:

$$\sum_{i=1}^{p} \lambda_i f_i(x) = \operatorname{Sup}_{y \in X} \sum_{i=1}^{p} \lambda_i f_i(y) \qquad (*);$$

alors $x \in P_f$

2) S'il existe des scalaires $\lambda_i (i=1, \ldots p)$ tous strictement positifs et tels que la relation (*) soit vérifiée, alors $x \in P$.

Preuve:

1) Soit $(\lambda_1, \ldots, \lambda_p) \in \mathbb{R}^p - \{0\}$.

Si $x \notin P_f$, il existe alors y tel que xPy. Nous avons:

$$\sum_{i=1}^{p} \lambda_i f_i(y) - \sum_{i=1}^{p} \lambda_i f(x) = \sum_{i=1}^{p} \lambda_i (f_i(y) - f_i(x)) > 0 \quad ;$$

ce qui est impossible puisque:

$$\sum_{i=1}^{p} \lambda_i f_i(x) \geq \sum_{i=1}^{p} \lambda_i f_i(y) \qquad \text{d'après (*).}$$

La démonstration dans le 2° cas est tout aussi évidente.

Application. Si x maximise l'un des critères sur X, x est un optimum de Pareto faible. Par contre, si x maximise, par

exemple, $\sum_{i=1}^{p} f_i$, $x \in P$.

Le théorème précédent signifie que, en particulier, toute

combinaison convexe des critères $\sum_{i=1}^{p} \lambda_i f_i$ constitue un critère

dont l'optimisation permet de sélectionner un optimum de Pareto faible. Bien entendu, le choix de la pondération $\lambda_1, \ldots, \lambda_p$ est conflictuel: si x maximise

$$\lambda_1 f_1 + \ldots + \lambda_p f_p$$

et si x' maximise
$(\lambda_1 + \varepsilon) f_1 + \lambda_2 f_2 + \ldots + \lambda_p f_p$ avec $\varepsilon > 0$, alors $f_1(x) \leq f_1(x')$.

En effet, nous avons à la fois:

$$\sum_{i=1}^{p} \lambda_i f_i(x') \leq \sum_{i=1}^{p} \lambda_i f_i(x)$$

et

$$\varepsilon f_1(x) + \sum_{i=1}^{p} \lambda_i f_i(x) \leq \varepsilon f_1(x') + \sum_{i=1}^{p} \lambda_i f_i(x')$$

en ajoutant membre à membre et en simplifiant, il vient:

$\varepsilon f_1(x) \leq \varepsilon f_1(x') \Leftrightarrow f_1(x) \leq f_i(x')$.

Ainsi, en augmentant le poids d'un critère, on ne peut qu'améliorer sa valeur aux optima de Pareto correspondants: on favorise ce critère.

Nous avons laissé dans l'obscurité un point intérssant qui est que nous pouvons toujours nous ramener à une combinaison convexe

$$(\text{i.e. } \sum_{i=1}^{p} \lambda_i = 1)$$

En effet, si x vérifie:

$$\sum_{i=1}^{p} \lambda_i f_i(x) = \sup_{y \in X} \sum_{i=1}^{p} \lambda_i f_i(y) \ ;$$

nous avons

$$\frac{1}{\sum \lambda i} \sum \lambda i \, fi(x) = \frac{1}{\sum \lambda i} \sup_{y \in X} \sum \lambda i \, fi(y) = \sup_{y \in X} \frac{1}{\sum \lambda i} \sum \lambda i \, fi(y).$$

2.8. Les fonctions d'utilité

Comme nous l'avons déjà indiqué plus haut, il est très rare qu'un point x de X maximise simultanément toutes les fonctions objectifs $f_1, ..., f_p$. Le théorème fondamental énoncé en (2.6) montre que certains types de compositions des fonctions $f_1, ..., f_p$ (on dit aussi des "agrégations" de telles fonctions), permettent, par maximisation de la fonction d'agrégation, l'obtention d'éléments de P.

Une fonction d'utilité est, par définition, une fonction $U : \mathbb{R}^p \to \mathbb{R}$. Afin que cette définition possède un intérêt pratique, des hypothèses sur la fonction U, sont faites. Généralement, des propriétés de concavité, de pseudo-concavité, de quasi-concavité, ... sont exigées pour une telle fonction. Ces propriétés sont sensées traduire mathématiquement des préférences considérées comme "naturelles" pour le décideur. La théorie de l'utilité en économie ayant fait l'objet de nombreux ouvrages, nous ne développerons pas davantage l'étude des fonctions d'utilité. Dans la suite, notre intérêt pour les fonctions d'utilité, repose sur la remarque suivante.

- Supposons que la fonction U, dépendant éventuellement de paramètres, soit choisie de telle sorte que les éléments de P appartiennent à l'ensemble des maxima de la fonction U; la recherche des éléments de P s'en trouve simplifiée (du moins en théorie).

Remarquons que la fonction décrite dans le théorème fondamental, est une fonction d'utilité particulière. Son intérêt ne s'arrête pas au résultat déjà cité, comme nous le verrons ultérieurement.

Donnons dès à présent, un cas d'application de la remarque précédente:

Théorème 1. Si la fonction $U : \mathbb{R}^p \to \mathbb{R}$ est strictement croissante par rapport à chaque coordonnée; alors, tout x qui maximise la fonction composée $U(f_1, ...,f_p)$,est un élément de P.

Preuve: Soit $X \supset \mathbb{R}^n$ le domaine de définition des fonctions $f_1, ...,f_p$; supposons que x maximise $U(f_1, ...,f_p)$ sur X. Si $x \notin P$, il existe $x^* \in X$ tel que, par définition:

$$f_1(x^*) \geq f_1(x), \ldots, f_p(x^*) \geq f_p(x);$$

l'une au moins des inégalités précédentes étant stricte.

Supposons, par exemple, que $f_i(x^*) > f_i(x)$. Comme U est strictement croissante par rapport à chacune de ses composantes,

$U(f_1(x^*),..., f_i(x^*),..., f_p(x^*1)) > U(f_1(x),..., f_i(x),..., f_p(x))$

ce qui contredit l'hypothèse faite sur x.

L'obtention des optima de Pareto-faible peut se faire, sous des hypothèses assez larges, à l'aide d'une fonction d'utilité standard déjà rencontrée dans le théorème fondamental. Avant d'énoncer une caractérisation de ces éléments, fixons quelques notations:

On note: $\mathbb{R}^p_+ = \{(\lambda_1, ..., \lambda_p) / \lambda_i \geq 0 \quad \forall i=1, ..., p\}$

$\mathbb{R}^p_- = \{(\nu_1, ..., \nu_p) / \nu_i \leq 0 \quad \forall i=1, ..., p\}$

$A = [f_1, ...,f_p](X) + \mathbb{R}^p_- = \{f_1(x), \ldots, f_p(x)) + (\nu_1, ..., \nu_p) /$

$x \in X, \ \nu_1 \leq 0, \ldots, \nu_p \leq 0\}.$

Notons alors le résultat suivant:

• $(\exists\, y \in X, xPy) \Leftrightarrow (f_1(x), ..., f_p(x)) \in \mathring{A}$ (intérieur de A).

Cette propriété résulte du fait que: $f_i(y) > f_i(x) \quad \forall i=1, ..., p$

Posons $\varepsilon_i = f_i(y) - f_i(x)$; le pavé

$$\left]\, f_1(x) - \frac{\varepsilon_1}{2}, f_1(x) = \frac{\varepsilon_1}{2}\right[\ldots \left[f_p(x) - \frac{\varepsilon_p}{2}, f_p(x)) + \frac{\varepsilon_p}{2}\right[$$

est contenu dans A.

Par conséquent,

• $x \in P_F \Leftrightarrow (f_1(x), ..., f_p(x)) \notin \mathring{A}$

Théorème 2. Soit X une partie convexe et soient $f_1, ..., f_p$, des fonctions concaves définies sur X. Un élément x de X est un Pareto faible si et seulement s'il existe des scalaires

$$\lambda_1 \geq 0, ..., \lambda_p \geq 0, \text{ tels que: } \sum_{i=1}^{p} \lambda_i f(x) = \sup_{y \in X} \sum_{i=1}^{p} \lambda_i f_i(y)$$

Preuve. Une partie de l'énoncé a fait l'objet du théorème fondamental. Montrons que si $x \in P_F$, il existe $\lambda_1 \geq 0, ..., \lambda_p \geq 0$.

Nous venons de voir que: $x \in P_F \Leftrightarrow (f_1(x), ..., f_p(x)) \notin \mathring{A}$;
d'autre part, nous savons que si X est convexe et les f_i concaves, A est un ensemble convexe.
Il s'ensuit que son intérieur $\mathring{A}$ est aussi un ensemble convexe.
D'après le théorème de séparation faible, il existe $\lambda_1, ..., \lambda_p$ non tous nuls vérifiant:

$$\sup_{a \in \mathring{A}} \sum_{i=1}^{p} \lambda_i a_i \leq \sum_{i=1}^{p} \lambda_i f_i(x) .$$

Supposons $\lambda_1 < 0$. En prenant $a_1 \to -\infty$ (ce qui est possible car $A = [f_1, ... f_p](X) + \mathbf{R}^p$) et en laissant les autres valeurs fixées, on obtient:

$$\sum_{i=1}^{p} \lambda_i a_i \to +\infty ;$$

cela est absurde car:

$$\sum_{i=1}^{p} \lambda_i a_i \leq \sup_{a \in \mathring{A}} \sum_{i=1}^{p} \lambda_i a_i \leq \sum_{i=1}^{p} \lambda_i f_i(x)$$

qui est fini; Il s'en suit que: $\lambda_1 \geq 0$. Nous montrerions de même que $\lambda_2, ..., \lambda_p$ sont tous ≥ 0.

Puisque $a \in \mathring{A}$, a s'écrit: $a = (f_1(y), ..., f_p(y)) + (\underset{\leq 0}{---}, ..., \underset{\leq 0}{---})$

Remarquons que:

$$\sup_{a \in A} \sum_{i=1}^{p} \lambda_i a_i = \sup_{a \in A} \sum_{i=1}^{p} \lambda_i a_i = \sup_{y \in X} \left[\sum_{i=1}^{p} \lambda_i f_i(y) + \sum_{i=1}^{p} \lambda_i v_i \right]$$
$$v_1, \ldots, v_p \leq 0$$

Supposons que:

$$\sup_{y \in X} \sum_{i=1}^{p} \lambda_i f_i(y) > \sum_{i=1}^{p} \lambda_i f_i(x) ;$$

il existe alors $y_0 \in X$ tel que:

$$\sum_{i=1}^{p} \lambda_i f_i(y_0) > \sum_{i=1}^{p} \lambda_i f_i(x) ;$$

En prenant des v_i tous nuls nous aurions:

$$\sum_{i=1}^{p} \lambda_i f_i(y_0) + \sum_{i=1}^{p} \lambda_i v_i > \sum_{i=1}^{p} \lambda_i f_i(x)$$

d'où, a fortiori :

$$\sup_{y \in X} \left[\sum_{i=1}^{p} \lambda_i f_i(y) + \sum_{i=1}^{p} \lambda_i v_i \right] > \sum_{i=1}^{p} \lambda_i f_i(x)$$
$$v_1, \ldots, v_p \leq 0$$

cette inégalité équivaut à:

$$\sup_{a \in A} \sum_{i=1}^{p} \lambda_i a_i > \sum_{i=1}^{p} \lambda_i f_i(x)$$

relation qui est contradictoire avec ce qui précède. Ainsi,

$$\sup_{y \in X} \sum_{i=1}^{p} \lambda_i f_i(y) \leq \sum_{i=1}^{p} \lambda_i f_i(x)$$

comme $x \in X$, on a bien l'égalité.

Remarques.

1) La concavité des critères f1, ... , fp n'entraîne pas la convexité de l'ensemble [f1, ... , fp](x) ; Mais seulement la convexité de l'ensemble:

$[f1, \ldots, fp](x) + \mathbf{R}^p$.

2) Le théorème 2 caractérise les optima de Pareto-faibles sous l'hypothèse de concavité. il n'est malheureusement pas possible d'obtenir une caractérisation analogue des optima de Pareto, nous souhaiterions un résultat du type:

$x \in P \Longleftrightarrow \exists\, \lambda_1 > 0, \ldots, \exists\, \lambda_p > 0$ vérifiant:

$$\sup_{a \in A} \sum_{i=1}^{p} \lambda_i a_i = \sum_{i=1}^{p} \lambda_i f_i(x) .$$

Il existe des contre-exemples montrant que cela est impossible. Toutefois, dans un cas particulier très important pour les applications, ce résultat est vrai.

Théorème 3. Si X est l'enveloppe convexe d'un nombre fini de points (X est un polytope) et si les critères f1, ... , fp sont affines sur X, alors:

$x \in P \Longleftrightarrow \exists\, \lambda_1 > 0, \ldots, \exists\, \lambda_p > 0$ vérifiant:

$$(*) \qquad \sum_{i=1}^{p} \lambda_i f_i(x) = \sup_{y \in X} \sum_{i=1}^{p} \lambda_i f_i(y)$$

Preuve .

1) S'il existe $\lambda 1 > 0, \ldots, \lambda p > 0$ vérifiant (*), le théorème fondamental affirme que x est optimum de Pareto.

2) Inversement, si x est un optimum de Pareto, montrons qu'il existe de tels scalaires $\lambda 1, \ldots, \lambda p$. Puisque X est un polytope, $A_0 = [\, f_1, \ldots, f_p \,](X)$ est aussi un polytope (mais de $\mathbf{R}^p$).

Appellons $a^1, \ldots, a^k$ les sommets de ce polytope A_0 .

$$A_0 = < a^1, \ldots, a^k > .$$

Soit:

$$\Lambda = \{\, \lambda \in \mathbf{R}^p_+ \;/\; \sup_{a \in Ao} \sum_{i=1}^{p} \lambda_i a_i = \sum_{i=1}^{p} \lambda_i f_i(x) \,\}$$

(Attention ces a_i ne sont pas les sommets de A_0). Λ est un cône convexe fermé non vide ($0 \in \Lambda$). Puisque $x \in P_F$, le cône Λ coupe $\mathbf{R}^p_+ - \{0\}$; Nous cherchons à montrer que Λ coupe :

$$\begin{matrix} 0 \\ \mathbf{R}^p_+ \end{matrix}$$

(i. e. n'est pas contenu dans une face de $\mathbf{R}^p_+$). Supposons le contraire, et donc que Λ soit contenu dans une face de $\mathbf{R}^p_+$. En conséquence :

(*) $\exists i \in \{1, \ldots, p\}$, $\forall \lambda \in \Lambda$, $\lambda i = 0$.

Or, Λ est défini par:

$$\Lambda = \{ \lambda \in \mathbf{R}^p_+ / \forall k = 1, \ldots, K , \sum_{i=1}^{p} \lambda_i a_i^k \leq \sum_{i=1}^{p} \lambda_i f_i(x) \}$$

(a_i^k représente la i ème composante de a^k). Cela résulte du fait que tout $a \in A_0$ s'écrit:

$$a = \sum_{k=1}^{K} \alpha_k a^k \quad \alpha_k \geq 0 \qquad \sum_{k=1}^{K} \alpha_k = 1 .$$

Notons $\alpha = (f1(x), \ldots, fp(x))$; il résulte de la relation (*) précédente que:

$\exists i \in \{ 1, \ldots, p \}$, $\forall \lambda \in \Lambda$, $\forall k = 1, \ldots, K$,

$$[\lambda, a_k - \alpha] \leq 0 \Rightarrow \lambda i = 0 .$$

(On utilise la nouvelle définition de Λ).

Appelons $\{ e^1, \ldots, e^p \}$ la base canonique de $\mathbf{R}^p$.

si $\lambda \in \mathbf{R}^p$ et si:

(**) $[\lambda , e^1] \geq 0 ; \ldots ; [\lambda , e^p] \geq 0 ; [\lambda , \alpha - a^1] \geq 0; \ldots$

$[\lambda , \alpha - a^K] \geq 0 ;$ alors $\lambda \in \Lambda$.

Par hypothèse, $\lambda i = 0$ et donc nous pouvons rajouter $[\lambda, -e^i] \geq 0$

D'après le théorème de Farkas nous savons que:

$$(F) \quad -e^i = \sum_{k=1}^{K} \theta_k (\alpha - a^k) + \sum_{j=1}^{p} \nu_j e^j$$

$$\theta_k \geq 0 \quad \forall k \; ; \quad \nu_j \geq 0 \quad \forall j .$$

Si $\theta_1 = \ldots = \theta_k = 0$, alors :

$$-e^i = \sum_{j=1}^{p} \nu_j e^j \text{ , donc}$$

$$-e^i - \nu_j e^i = \sum_{j \neq i} \nu_j e^j \text{ , soit encore}$$

$$\sum_{j \neq i} \nu_j e^j + e^i (1 + \nu_i) = 0 .$$

Les e^j constituant une base, il s'en suit que: $\nu_j = 0$ si $j \neq i$ et $\nu_i = -1$ ce qui est absurde.

par conséquent, $\sum_{k=1}^{K} \theta_k > 0$.

En transformant l'égalité (F), il vient:

$$\sum_{k=1}^{K} \theta_k \, a^k = (\sum_{k=1}^{K} \theta_k)\alpha + e^i + \sum_{k=1}^{K} e^j .$$

Posons: $\theta_k^+ = \theta_k \Big/ \sum_{k=1}^{K} \theta_k$;

$$a = \sum_{k=1}^{K} \theta_k^+ \, a^k = \alpha + (1 / \sum_{k=1}^{K} \theta_k)\, e^i + \sum_{j=1}^{p} (\nu_j / \sum \theta_k)\, e^j .$$

Or $a \in A_0$ et vérifie :

$$a_j = f_j(x) + \frac{\nu j}{\sum \theta k} \geq f_j(x) \qquad (j \neq i)$$

$$a_i = f_i(x) + \frac{1}{\sum \theta_k} + \frac{\nu j}{\sum \theta_k} > f_i(x) .$$

On améliore donc α ; ce qui est en contradiction avec le fait que $x \in P$. Λ n'est donc pas contenu dans une face de $\mathbf{R}^p_+$. Il existe par conséquent, λ tel que $\lambda_i > 0$, $\forall$ i = 1, ... , p, et vérifiant de plus:

$$\sum_{i=1}^{p} \lambda i \, fi(x) = \sup_{y \in X} \sum_{i=1}^{p} \lambda i \, fi(y) .$$

2.8. Caractérisation des optima de Pareto dans l'ensemble image.

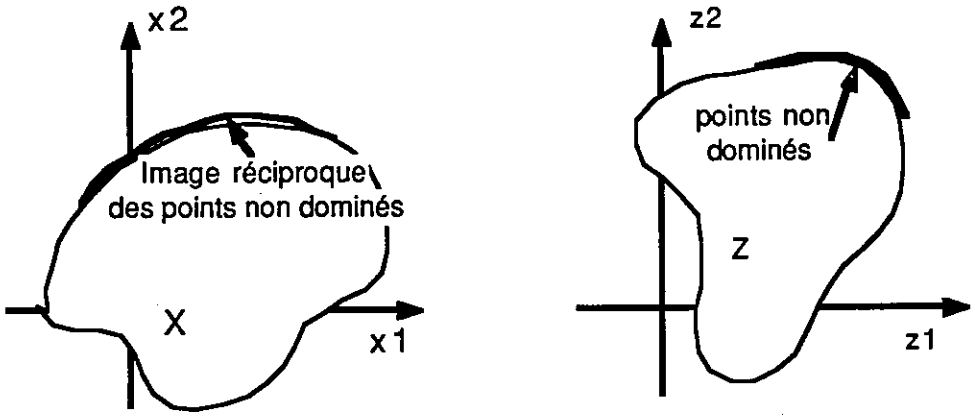
Nous noterons désormais $Z = [f_1, \ldots , f_p] (X)$ l'image de X par $(f_1, \ldots , f_p)$.

. Un point z^* de Z est dit non-dominé, s'il n'existe pas z de Z tel que $z \geq z^*$ et $z \neq z^*$. Dans le cas contraire, il est dit dominé.

Théorème . $x^* \in P$ si et seulement si $(f_1, \ldots , f_p)(x) = z^*$ est un point non dominé de Z.

La preuve est immédiate.

En d'autres termes, les éléments de P sont les images réciproques des points non-dominés de Z. Or, ces derniers, dans le cas de $\mathbf{R}^2$ sont aisément détectables, comme le montre l'illustration suivante. Nous disposons donc d'un procédé géométrique permettant de visualiser, parfois, dans un des cas simples, les optima de Pareto. Dans l'exemple peoposé, X est une partie de $\mathbf{R}^2$ et p=2.



LA MACHINE A INSTRUCTION UNIQUE

E. Bianco

Résumé.

On imagine une machine qui disposerait d'une seule instruction pour effectuer tous les calculs en mémoire centrale qu'on est censés attendre d'une machine complète. La comparaison avec la machine de Nolin donne un autre aspect à ce qu'on appelle le calcul automatique, et montre ce qu'on gagne effectivement en commodité de programmation en diversifiant les opérations élémentaires.

LA MACHINE à INSTRUCTION UNIQUE.

Au rayon des monstruosités on peut encore ajouter une pièce. La machine de Nolin montrait qu'avec quatre ou cinq instructions il est possible de construire un véritable ordinateur, au point qu'il est tout à fait raisonnable de la considérer comme une sorte de machine de référence, qui a l'allure des ordinateurs les plus complexes mais qui, néanmoins conserve une simplicité suffisante pour donner prise commode au raisonnement.

Je vais envisager une construction qui ne comporte qu'une seule instruction, et on va vérifier qu'en fait, cette machine est aussi puissante que la machine de Nolin. Pour faire bonne mesure, je la démunirai également de l'un de ses attributs qui peut paraître important: la case w. Quand je dis "machine à une instruction" il faut entendre par là: une seule instruction pour travailler en mémoire centrale. Je laisse de côté les files externes, indispensables bien entendu, mais qui n'entrent pas en ligne de compte dans cette théorie.

La démonstration, de forme classique désormais, consiste à prouver qu'il n'existe pas d'algorithme de la machine de Nolin à qui on ne puisse pas faire correspondre au moins un algorithme de la machine à une instruction. La réciproque ne présente aucune difficulté.

Je commence par définir une "mémoire centrale" constituée d'une suite de cases en nombre k, toutes identiques entre elles, et ne différant que par leurs noms qui permettent de les atteindre. Je montrerai que je n'ai pas besoin de case spécialisée du type case W. Les contenus des cases sont à prendre dans une liste prédéfinie d'objets tous différents les uns des autres. Pour reprendre une terminologie déja traditionnelle, je dirai que le contenu de chacune des cases peut être rien d'autre que l'une quelconque des lettres qui appartiennent à un alphabet fini, donné.

Je dirai que les cases se nomment:

0, 1, 2, 3, 4, ... , (k-1)

et que leur contenu est l'une des lettres:

A = { x1, x2, x3, ..., xn }.

L'instruction unique s'écrit:

I si ca = xi alors ca := xj vers E ;

qui signifie que si l'on constate que le contenu de la case désignée, ici la case a, est la lettre "xi" alors on remplace xi par xj. Les lettres xi et xj appartiennent à A et peuvent être identiques ou différentes.

L'instruction à dérouler ensuite est celle qui porte l'étiquette E, première partie de commutation.

Si le contenu de a n'est pas xi, alors se déroule l'instruction suivante.

La notion d'algorithme est classique: un algorithme commence par la parenthèse début et s'achève par l'autre parenthèse fin. Entre, se trouve une suite quelconque, au gré du programmeur, d'instructions du type I . Certaines d'entre elles peuvent être étiquetées ce qui représente la seconde partie de commutation.

Tout algorithme de la machine de Nolin est constitué d'une suite quelconque d'objets dont la forme est à choisir entre les formes suivantes:

J1 ca := 0
J2 ca := cb
J3 ca := ca $+_K$ 1
J4 si ca = 0 vers F
J5 vers G

Je pars donc avec une machine de Nolin dont la mémoire possède m cases numérotées de 0 à m. Je me donne une machine à instruction unique dont la mémoire comporte k cases avec k ≥ m. Et je vais établir une correspondance entre chacun des contenus possibles des deux machines, arbitrairement si les contenus de cases de la machines de Nolin sont les entiers de 0 à (K-1) , je fais correspondre chacune de ces valeurs à une lettre de l'autre machine:

0:x1, 1:x2, 2:x3, ... , (K-1):xn,

ce qui impose de toute évidence n=K.

Je vais faire jouer à x1 le rôle du zéro, et faire en sorte que les lettres soient ordonnées comme le sont les entiers auquels elles correspondent. Je désigne par MIU la machine à instruction unique.

L'instruction J1.

Chaque fois que je rencontre l'instruction ca := 0, cela signifie que la valeur 0 est imposée dans la case a quel qu'ait pu être son contenu préalable. En MIU, cela revient à imposer la lettre x1, cela s'écrit:

si ca = x1 alors ca := x1 vers Z
si ca = x2 alors ca := x1 vers Z
si ca = x3 alors ca := x1 vers Z
- - - - - - - - - - - - - -
si ca = xn alors ca := x1 vers Z
Z: (instruction suivante)

On s'aperçoit que quel que soit le contenu de a, cet algorithme le remplace par x1 qui correspond au zéro.

L'instruction J2 .

Rencontrer ca := cb signifie qu'on remplace le contenu de a par celui de la case b. Il me suffit en MIU de sonder chaque valeur possible de b et de la transferer dès que je l'ai trouvée:

si cb = x1 alors cb := x1 vers P1
si cb = x2 alors cb := x2 vers P2
si cb = x3 alors cb := x3 vers P3
- - - - - - - - - - - - - - -
si cb = xn alors cb := xn vers Pn

P1 : si ca = x1 alors ca := x1 vers P
si ca = x2 alors ca := x1 vers P
si ca = x3 alors ca := x1 vers P
- - - - - - - - - - - - - - -
si ca = xn alors ca := x1 vers P

P2 : si ca = x1 alors ca := x2 vers P
si ca = x2 alors ca := x2 vers P
- - - - - - - - - - - - - - -
si ca = xn alors ca := x2 vers P

P3 : si ca = x1 alors ca := x3 vers P
si ca = x2 alors ca := x3 vers P
- - - - - - - - - - - - - - -
si ca = xn alors ca := x3 vers P

- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -

Pn : si ca = x1 alors ca := xn vers P
si ca = x2 alors ca := xn vers P
- - - - - - - - - - - - - - -
si ca = xn alors ca := xn vers P

P : (instruction suivante)

Là encore, je suis obligé d'explorer tous les contenus possibles de la case b pour transférer celui qui s'y trouve effectivement dans la case a.

On aperçoit la simplification importante de cet algorithme si on permet dans l'instruction unique de travailler sur deux cases différentes. En ce cas la traduction de la projection devient:

si cb = x1 alors ca := x1 vers P
si cb = x2 alors ca := x2 vers P
si cb = x3 alors ca := x3 vers P
- - - - - - - - - - - - - - -
si cb = xn alors ca := xn vers P

P : (instruction suivante)

L'instruction J3 .

C'est la seule instruction qui transforme une valeur, de par la correspondance établie plus haut, rencontrer x1 revient à le remplacer par x2, rencontrer x2 à le remplacer par x3, etc, rencontrer xn à le remplacer par x1, en même temps il faut noter, soit x1 quelquepart dans les premiers cas soit x2 dans le dernier. Pour noter l'équivalent du débordement j'utiliserai une case quelconque, par exemple la dernière de la mémoire. Ceci implique, si je veux pouvoir représenter complètement le contenu de mémoire de la machine de Nolin dans la mémoire de la MIU, que cette dernière possède une case de plus. Donc:

$$k = m+1 .$$

C'est la case (k-1) qui contient en conséquence le débordement.

si c(k-1) = x1 alors ca := x1 vers S3 Mise à zéro au préalable.
si c(k-1) = x2 alors ca := x1 vers S3
si c(k-1) = x3 alors ca := x1 vers S3
- - - - - - - - - - - - - - -
si c(k-1) = xn alors ca := x1 vers S3

S3 : si ca = x1 alors ca := x2 vers S1
si ca = x2 alors ca := x3 vers S1
si ca = x3 alors ca := x4 vers S1
- - - - - - - - - - - - - - - -
si ca = x(n-1) alors ca := xn vers S1
si ca = xn alors ca := x1 vers S2

S2 : si c(k-1) = x1 alors ca := x2 vers S1
S1 : (instruction suivante)

Donc toujours le principe de base, explorer toutes les possibilités de contenus de la case sur laquelle on travaille.

L'instruction J4 .

La comparaison est la plus simple a simuler, en effet:

si ca = x1 alors ca := x1 vers F

on reprend ici la même étiquette que celle qu'on trouve dans l'instruction de Nolin. Il faut faire ici une remarque à propos des étiquettes que j'utilise dans les présents algorithmes de la MIU. L'algorithme de la machine de Nolin dont on part, a priori quelconque, contient déja des étiquettes, comme F par exemple ou G. Il faut veiller à ce que des étiquettes comme Z, P, S3, S2, S1 en soient différentes.

L'instruction J5 .

On travaille sur n'importe quelle case, par exemple c0:

si c0 = x1 alors ca := x1 vers G
si c0 = x2 alors ca := x2 vers G
si c0 = x3 alors ca := x3 vers G
- - - - - - - - - - - - - -
si c0 = xn alors ca := xn vers G

Conclusion .

On se donne une machine de Nolin quelconque, partant d'une configuration quelconque, représentée par un contenu particulier de la mémoire centrale, après application de l'algorithme on aboutit à un nouveau contenu de mémoire qui représente la configuration résultat. Si je transforme l'algorithme en utilisant les formes exposées ci-dessus, sans oublier de respecter les précautions qui concernent les étiquettes, si je constitue un contenu de mémoire de la MIU en plaçant dans chaque case homologue de la machine de Nolin la lettre qui correspond à l'entier qui y est contenu, alors si je lance le calcul je vais obtenir une configuration résultat dans la mémoire de la MIU. Si dans cette configuration, on remplace chaque lettre par l'entier qui lui correspond on on obtient une configuration identique à la configuration résultat de la machine de Nolin. C'est en cela qu'on dit que ces deux machines ont même puissance.

La MIU est de toute évidence voisine de la machine de Turing pour ce qui est des actes opératoires élémentaires, en effet, on lit un contenu de case puis après l'avoir reconnu on écrit une lettre, la même ou une autre, dans cette même case. Par contre, le choix de la case est direct, et non pas calculé comme c'est le cas en machine de Turing.

Visiblement on peut faire disparaître la case w sans dommage, certes une case standard la remplace pour contenir le débordement, mais ce qui est important c'est qu'on a fait disparaître le mécanisme interne dont le rôle était précisément de faire évoluer le contenu de w. Si je fais jouer ce rôle à la case (k-1) , c'est pour pouvoir traduire des instructions de la forme:

si cw = 0 vers X
cw := ca
ca := cw

Bien entendu on doit conserver les deux instructions d'échanges avec la ou les files externes ainsi que les deux instructions de déplacement de cases sur ces files. On peut discuter des valeurs relatives de n le nombre de lettres de l'alphabet de la MIU, avec K le contenu maximum d'une case de la machine, de Nolin. J'ai supposé que n= K. Si n était plus grand il suffirait de laisser de côté les lettres non indispensables. Dans le cas d'un n trop petit il est encore possible de trouver une solution en attribuant par exemple deux cases de la MIU pour une de la machine de Nolin, et en faisant correspondre à chaque entier une combinaison différente de deux lettres. Les algorithmes de la MIU se compliqueraient un peu, car ils devraient travailler sur deux cases au lieu d'une à la fois. La MIU devrait disposer alors de 2m + 1 cases.

Réciproque .

Une instruction quelconque si ca = xi alors ca := xj vers P se traduit en machine de Nolin par une recherche de la valeur qui correspond à xi soit l'entier (i-1) et à la reconstruction d'une valeur équivalente à xj, soit l'entier (j-1).

Je vais le montrer sur un exemple particulier:

si ca = x3 alors ca := x5 vers P

en tenant compte des correspondances de x3 à 4 et de x5 à 6, je suppose de plus que K=9 , ce qui me donne droit aux nombres:

0, 1, 2, 3, 4, 5, 6, 7, 8

pour savoir s'il y a 4 il suffit d'ajouter 5 et de contrôler le débordement:

```
    ca := ca +K 1
    ca := ca +K 1
    ca := ca +K 1
    ca := ca +K 1
    ca := ca +K 1
    si ca = 0 vers E1
    vers SUITE
E1 : ca := 0
    ca := ca +K 1
    ca := ca +K 1
    ca := ca +K 1
```

ca := ca $+_K$ 1

ca := ca $+_K$ 1

ca := ca $+_K$ 1

SUITE : (instruction suivante)

Il est clair que ce qu'on sait faire pour ces deux lettres on sait le faire pour tout autre couple. De la même manière, ce qu'on sait faire pour K=9, on sait également le faire pour toute autre valeur de K. Donc il n'existe pas d'algorithme de la MIU qui n'ait pas au moins un équivalent en terme de machine de Nolin.

Amélioration de l'algorithme de production

par

BOUDIBA Ennacer

Résumé:

L'idée principale de l'article est l'étude du système formel (le calcul propositionnel) qui a pour objectif de rendre le calcul propositionnel directement utilisable du point de vue informatique (Développement des applications). Citons par exemple l'intérêt de cette étude pour les systèmes experts: organisation et programmation des bases de connaissances en calcul propositionnel . Cette étude est basée sur l'amélioration des algorithmes existants avec présentation de nouveaux algorithmes .

INTRODUCTION

Le travail réalisé au niveau de cet article est celui de l'amélioration de l'algorithme de 'PRODUCTION' (champs de production) < **Pierre SIGEL 1987** > , qui consiste à étudier dans le détail l'amélioration réalisée sur l'algorithme ' **SL-RESOLUTION** ' <C.CUBADDA & M.D.MOUSSEIGNE> et de voir s'il est possible de l'appliquer sur l'algorithme de 'PRODUCTION' et d'en tirer les conséquences .

Le travail est présenté de la façon suivante :

-Présentation des outils de travail :

. Le raffinement de **C.CUBBADA & M.D.MOUSSEIGNE** .

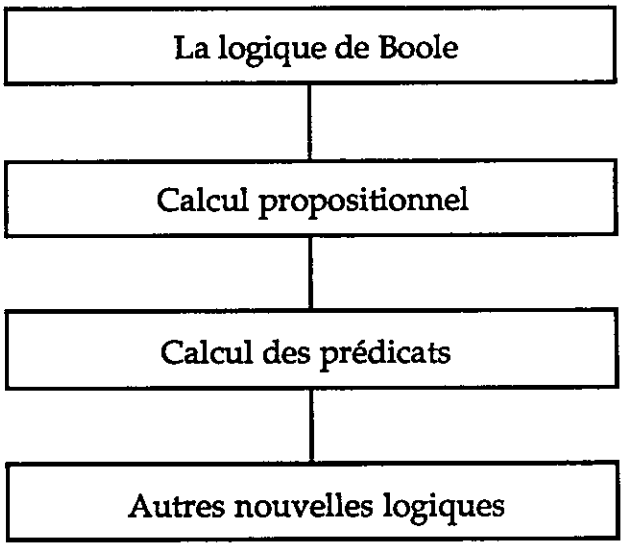
. L'algorithme de **PRODUCTION** .

-Formalisation des résultats avec validation.

-Présentation du nouvel algorithme .

-Les résultats Pratiques .

Evolution de la logique symbolique :



definition du calcul propositionnel : Le calcul des propositions ne s'occupe pas du problème le plus difficile qui est l'analyse des propositions , il prend les propositions globales non analysées pour les manipuler et les combinerau moyen de diverses opérations logiques :

. Proposition (expression).
. Conjonction (p et q).
. Disjonction (p ou q) .
. Négation (non q).
. Implication (p =====> q).
. Règle de détachement (modus ponens)
. Equivalence logique (p = q).

Principe de base de l'Intelligence Artificielle :

Le principe de base de l'intelligence artificielle est celui de La représentation des connaissances. Ces connaissances peuvent s'exprimer de plusieurs manières que ce soit au moyen du francais , des graphiques , des mathématiques , des règles de production (ou clauses de Horn) , **clauses** ,..... etc , et à partir de ces connaissances **on déduira** d'autres connaissances nouvelles .

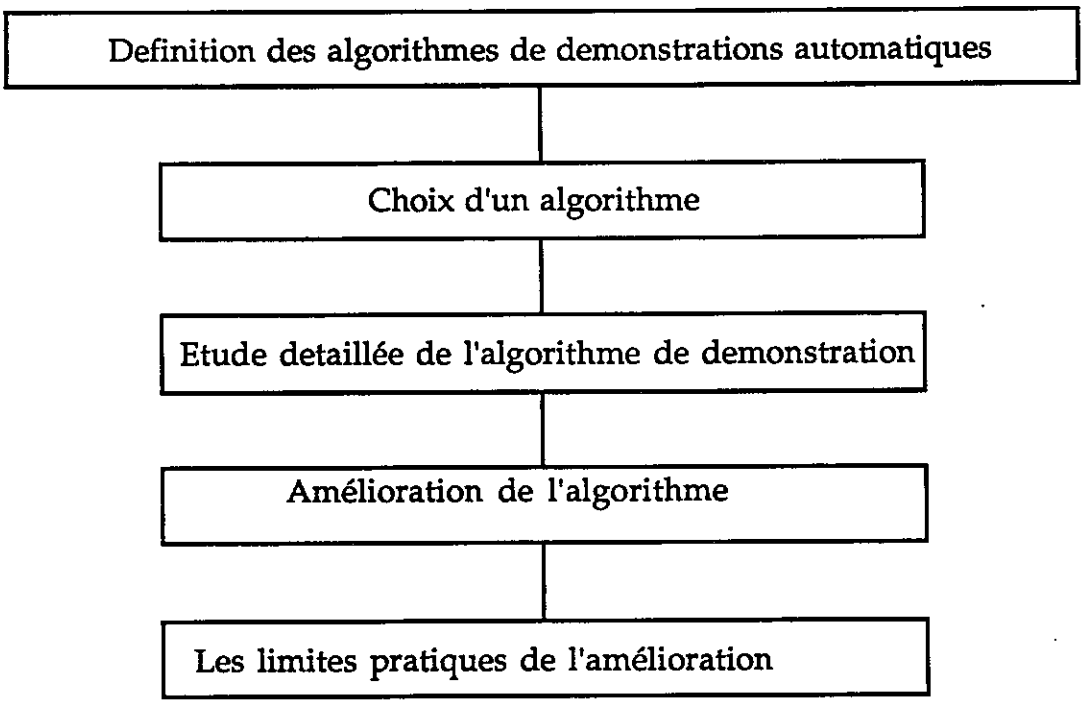
Parmis les axes de recherches du domaine de l'intelligence artificielle se trouve **l'utilisation pratique** de la logique , cela veut dire que la **déduction** des nouvelles connaissances se réalise à l'aide de programmes qui devront donner des résultats en un temps raisonnable; d'où l'apparition de la notion **d'algorithme de démonstration automatique en intelligence artificielle .**

Remarque importante :

Le travail qui va suivre se limite à l'utilisation de la logique du calcul propositionnel .

La représentation utilisée dans ce travail est celle de la représentation clausale.

Démarche utilisée :



Définitions des algorithmes de démonstration automatique:

Les algorithmes de démonstration automatique utilisés dans le calcul propositionnel qu'ils soient syntaxiques ou sémantiques ont été étudiés .

Parmi les algorithmes syntaxiques :

-SL_RESOLUTION de R.Kowalski & D.Kuner(1971).
-SATURATION
-PRODUCTION(Champ de production) Pierre SIEGEL (1987)

Parmi les algorithmes sémantiques :

- La procedure de Davis & Putnam (1960)

Objectif du travail :

L'algorithme de démonstration automatique la **SL_RESOLUTION** de R.Kowalski et D.Kuner(1971) présente un inconvénient majeur qui est la génération redondante et inutile des informations . Compte tenu de cet inconvénient cet algorithme n'offre que la possibilité de résoudre des problèmes simples avec un temps de réponse bien plus long que pour d'autres algorithmes existants .Des travaux d'amélioration effectués sur cet algorithme ont donné naissance à un nouvel algorithme appelé **SLRI**(SL_RESOLUTION avec Remontée d'Impasses) mis au point par le binôme C.CUBADDA & M.D.MOUSSEIGNE(1988).

L'idée principale de cet article est de prendre le principe d'amélioration de l'algorithme de R.Kowalski et de l'appliquer sur l'algorithme de PRODUCTION (Champ de production) de Pierre Siegel (1987).

Terminologie utilisée:

Intelligence Artificielle:

le sous-champ de l'informatique qui s 'intéresse aux concepts et aux méthodes d'inférence symbolique pour un ordinateur et à la représentation symbolique du savoir à utiliser pour les inférences . Le champ qui vise à conférer à l'ordinateur un comportement reconnu comme intelligent par l'homme (Feigenbaum et Mc.Corduck,1983);

SL_RESOLUTION:

Algorithme de démonstration automatique de R.Kowalski & D.Kuner (1971).

PRODUCTION (Champ de production):

Algorithme de démonstration automatique de Pierre SIEGEL (1987)

Littéral ou proposition:

Variable propositionnelle a,b,.... (propositions positives) ainsi que leurs négations -a ,-b ,.....(propositions négatives) l'opposé d'une proposition positive a est -a ,celui d'une proposition négative -a est a .

Clause :

Une clause est une disjonction de littéraux.

Un graphe :

Un Graphe est un couple (N,A) où N est un ensemble de noeuds de A,inclus dans N*N , un ensemble d'arcs .

Chemin:

C = (x1,.....,xp) d'un noeud m à un noeud n est un élement de A^p tel que x1 correspond au noeud m et xp correspond au noeud n et tel que ,pour tout i qui appartient à {1,.....,p-1}, on ait **xi =(x,y) et xi+1 = (y,z) .**

Arbre :

Un arbre est un couple (G,o) où G est un graphe et o est un noeud tels que, pour tout m différent de o , on ait un unique chemin de o à m .

Formule:

Les variables propositionnelles sont des formules propositionnelles si f est une formule propositionnelle , -f en est une .

Si f1,f2,.....,fn sont des formules propositionnelles (n>1) alors

(f1 ∧ f2 ∧ ..... ∧ fn) et (f1 V f2 V f3 V V fn)

sont aussi des formules propositionnelles .

Tautologie:

Une tautologie est une clause contenant un littéral et son opposé.

Système formel :

Ensemble de données purement abstrait sans lien avec l'extérieur qui décrit les règles de manipulation d'un ensemble de symboles traités de façon uniquement syntaxique c'est à dire sans considération du sens . L'intérêt d'un système formel réside dans la possibilité de modéliser plusieurs situations différentes .

Decidabilité :

Existence d'un procédé bien déterminé qui en un nombre fini d'étapes donne un résultat . Le système formel associé est dit décidable .
ex: L'algorithme de démonstration automatique de **Production** est décidable (c.a.d que l'algorithme s'arrête car un ensemble de clauses fini donne un ensemble fini de productions).

Cohérence :

Les resultats sont cohérents veut dire que les résultats sont valides la cohérence au niveau du calcul propositionnel est que toute clause produite par une production de C est impliquée par C en particulier , si une production de C produit la clause vide, alors C est inconsistant.

Interprétation :

Mise en rapport du système formel avec l'univers , elle donne un sens à tout symbole du système formel en établissant une correspondance univoque entre les symboles du système formel et certains des objets de l'univers. L'interprétation au niveau du système formel (le calcul propositionnel) est une application qui associe à toute formule propositionnelle f une valeur **I[f]** dans **{0,1}**.

Système expert:

Un système expert peut être considéré comme un ensemble de connaissances de raisonnements et d'interfaces destiné à résoudre un problème particulier. La définition formelle d'un système expert est composée de trois parties principales (base de faits , base de connaissances, moteur d'inférences) qui utilisent les mêmes élements de connaissances appelés ' faits ' .

Moteur d'inférences:

Partie du système expert qui contient les stratégies de contrôle et d'inférence.Plus généralement , il contient également les différents modules d'acquisition du savoir ,d'explication et d'interface utilisateur .Les moteurs d'inférences sont caractérisés par les stratégies d'inférence et de contrôle qu'ils utilisent .Par exemple, le moteur d'inférence de MYCIN utilise le modus ponens et le chaînage arrière .

Base de connaissances :

Partie du système expert qui contient les faits et les heuristiques du domaine traité .

Heuristique :

Règle empirique , procédé ou simplification qui permet de réduire ou de limiter l'exploration des grands espaces de problème .

Contrairement aux algorithmes , les heuristiques ne garantissent pas des solutions correctes .

Base de faits (base de connaissance dynamique , mémoire a court-terme) :
La collection des informations touchant un utilisateur , qui représente une mémoire temporaire pour le système expert .Elle est mis à jour par la progression du raisonnement et vidée lorsqu'une consultation se termine.

Résolvante:
Une résolvante est une clause obtenue à partir de deux clauses (ces deux clauses pouvant être elles même des résolvantes) C1 = a1,....,an et C2 = b1.....bm telles qu'il existe un unique couple (i,j) avec i appartenant à (1,.....,n) et j apprtenant à (1.....m) tel que ai = -bj .La résolvante R(C1,C2) sera la clause (a1....ai-1 ai+1.....an) U (b1.....bj-1 bj+1.....bm) .

Backtraking (remontée de l'arbre) :
Remontée en arrière (Exemple : remontée dans l'arbre de production).

Présentation détaillée de la SL RESOLUTION :

But de l'algorithme:

L'algorithme de la Sl_RESOLUTION permet de déterminer si le système de clauses est consistant ou non.En effet si une clause ou deux littéraux opposés sont effacés avec un ensemble d'ancêtres A vide , alors le système est inconsistant

Principe du raisonnement :

Si p est un littéral et si E est un ensemble cohérent de clauses ,pour montrer que: E === p
On essaie d'éffacer -p à partir de E .Pour cela on cherche dans E une clause de la forme **p v x1 v x2 v xn**
et on produit une première résolvante : **x1 v x2 v xn**
Le principe consiste alors à effacer chaque littéral **xi** de la resolvante à partir de **EU{-p}** On dit que **-p** est un ancêtre de chaque littéral **xi**.
On continue ainsi jusqu'à ce qu'il ne reste plus rien à éffacer : dans ce cas **p** est vrai.

On généralise :
Effacer une clause consiste à effacer tous les littéraux de la clause.
En logique propositionnelle , on est sûr de s'arrêter si on n'essaie pas d'éffacer un littéral qui appartient à sa propre liste d'ancêtres .

Remarque :

Au niveau du moteur d'inférences des systèmes experts la SL-RESOLUTION représente l'algorithme utilisé sous le mode de chaînage avant (c'est à dire que le but est défini et que l'on tente de le vérifier à partir d'une base de connaissances et d'une base de faits).

Présentation détaillée de la SL RESOLUTION avec remontée d'Impasses:

Les variantes d'amélioration utilisées :

Les deux variantes utilisées pour l'amélioration de l'algorithme de la SL_RESOLUTION sont :

Variante de généralisation des B-ancêtres :

Au niveau de la variante des B-ancêtres , l'ensemble des littéraux effacés dans une clause c est perdu si celle ci conduit à un échec. La variante de généralisation des B-ancêtres consiste à conserver cet ensemble pour les tentatives d'effacement des clauses de même niveau que c par la variante des B-ancêtres.

Variante de remontée des impasses :

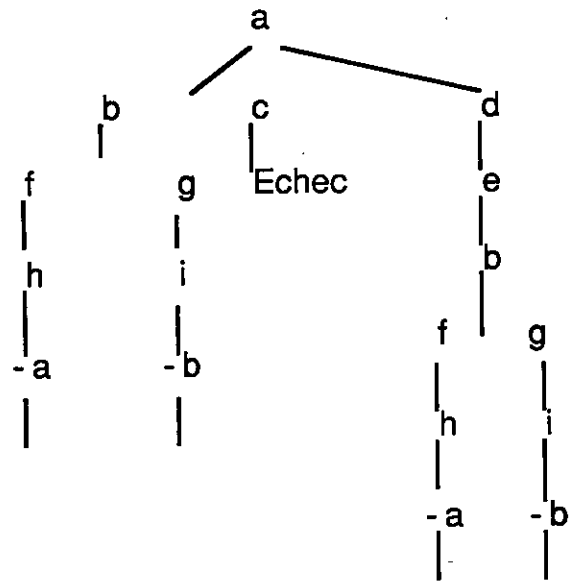
Cette variante consiste à mémoriser les informations des échecs précédents chaque fois qu'un littéral conduit à un échec . C'est à dire que le littéral l est ajouté à l'ensemble des I-ancêtres si le littéral l n'est pas éffacé et l'ensemble des I-ancêtres est réinitialisé à l'ensemble vide sinon.

Exemple :

On essaye de verifier à partir d'un ensemble E de clauses l'effacement d'un litteral donné .

on suppose que le litteral a effacer est : a

E={ (-a,b,c) , (-a,d) , (-b,f,g) , (-f,h) , (-g,i) , (a,-h) , (-b,-i) , (-d,e) , (b,-e) } .

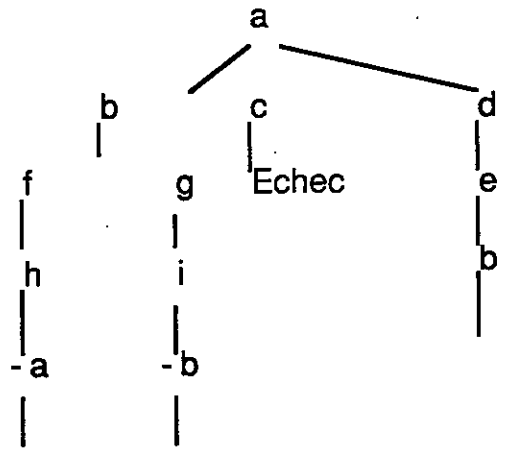


On commence par l'effaçement du littéral **a** , la clause **(-a,b,c)** contient le littéral opposé de **a** donc elle est choisie parmi l'ensemble des clauses .
-la clause **(-a,b,c)** devient la clause courante du processus , le littéral courant devient **b** ,recherche parmi l'ensemble des clauses s'il existe une clause contenant l'opposé du litteral **b** et qui verifie les conditions de l'algorithme, la clause selectionnée est la clause **(-b,f,g)**.
-La clause **(-b,f,g)** devient la clause courante et f devient le littéral courant .
recherche parmi l'ensemble des clauses l'existance d'un clause contenant l'opposé du littéral **f** et qui verifie les conditions de l'algorithme .
-La clause selectionnée est **(-f,h)**, la clause **(-f,h)** devient la clause courante et le littéral **h** devient le littéral courant .
-Recherche parmi l'ensemble des clauses l'existance d'une clause contenant l'opposé du littéral **h** et qui verifie les conditions de l'algorithme .
-La clause séléctionnée est **(-a,-h)** , le littéral courant devient **-a** et comme le littéral **a** est effaçable alors son opposé l'est aussi .
Ces étapes representent le traitement pour un littéral des clauses selectionnées,ainsi pour le trraitement des autres littéraux des autes clauses il suffit de remonter la branche traitée jusqu'au littéral de la premiere clause non traité et de refaire le même processus de traitement .

Remarque : Le symbole (Δ) presenté au niveau des arbres représente l'effacement d'un littéral .

Application de la variante de généralisation des B-ancêtres sur l'exemple :

Le résultat en utilisant la variante de généralisation des B-anêtres est le suivant , après la remontée de l'arbre le contenu du registre de sauvegarde est le littéral **b** (c'est à dire que le littéral **b** est effaçable).

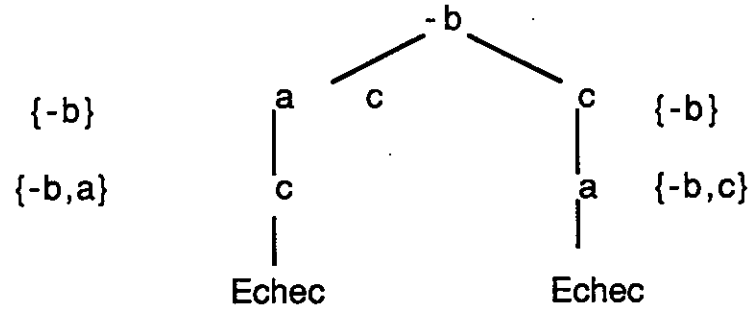


Le but de cette variante est de sauvegarder les littéraux qui conduisent a un effacement , en suivant le processus de traitement , on remarque que les feuilles de la branche sous la racine **b** sont representés par des littéraux effaçables donc on peut conclure que toute la branche est effacable a partir de la racine **b** ,ainsi le littéral **b** sera sauvegardé dans le registre de sauvegarde des litteraux effacables .

-On poursuit le processus de traitement , comme le dernier littéral de la clause selectionnée pour l'effacement du littéral **b** conduit a un echec (c.a.d n'y a pas une clause parmi l'ensemble des clauses contenant l'opposé du littéral **b**) alors on conclu que la premiere clause selectionnée ne conduit pas a un effacement , Alors il faut chercher parmi l'ensemble des clauses qui reste s'il existe une autre clause contenant l'opposé du littéral à effacer , la clause séléctionnée est la clause (**-a d**),**<<voir arbre ci-dessus la branche droite>>** le littéral courant devient **d** .
-Recherche parmi l'ensemble des clauses , l'existance d'une clause contenant l'opposé du littéral **d** alors la clause séléctionnée est la clause (**-d e**) , passage au littéral suivant de la clause **e** , qui va devenir le littéral courant .
-Recherche parmi l'ensemble des clause l'existance d'une clause contenant l'opposé du littéral **e**. La clause séléctionnée et qui vérifie les conditions de l'algorithme est (**b e**) , passage au littéral suivant qui est **b** .
-Comme on sait que dans le registre de sauvegarde se trouve le littéral **b** qui conduit a un effacement alors on affecte l'effacement pour le littéral courant qui est le **b** sans prendre en compte le parcours de l'effacement .
-En conclusion , on a eviter de parcourir une branche de l'arbre total d'effacement .

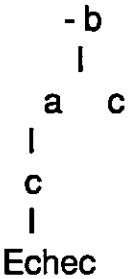
<u>Application de la variante de remontée des impasses sur l'exemple</u>:

Soit le système de clauses **E** suivant :
E = (a,b,c) , (b,c) , (a,-c) , (-a , c)
Arbre de recherche de l'algorithme de la SL_RESOLUTION est :



On veut effacer le littéral **-b** avec l'ensemble des clauses E:
-Recherche parmi l'ensemble des clauses l'existance d'une clause contenant l'opposé du littéral **-b** , la clause selectionnée est **(a b c)** et le littéral **a** devient le littéral courant .
- Recherche parmi l'ensemble des clauses E l'existance d'une clause contenant l'opposé du littéral **a** alors la clause **(-a c)** est la clause selectionnée , le littéral **c** devient le littéral courant .
- Pour le littéral **c** , n'y as pas une clause parmi l'ensemble des clauses contenant l'opposé du littéral **c** donc **c** conduit un échec ,alors la clause origine pour cet exemple **(-b a c)** conduit a un échec . En remontant la branche traitée
on arrive au littéral initiale **-b**.

Registre de sauvegarde :
On mémorise les littéraux **c ,a , " b"** conduit à un échec
Le nouvel arbre de recherche aprés application de la variante de remontée d'impasse est le suivant :



Le point fort en appliquant la variante de remontée des impasses (des échecs) est : comme le littéral **c** conduit a un échec descendant de la clause **(-b a c)** ,donc le littéral **a** conduit à un échec , ce dernier est descendant du littéral **-b** (la racine) alors ce littéral conduit aussi a un échec . D'ou le contenu du registre de sauvegarde sera le littéral **(-b)**.
D'aprés le processus de traitement il y a autres choix pour le littéral à effacer **-b** c.a.d qu'il existe parmi l'ensemble des clauses une clause autre que celle deja choisie contenant l'opposé de -b . En continuant le processus de traitement on tombe sur un échec .

L'amélioration se réside dans le faite que le registre de sauvegarde contient des littéraux conduisant a un échec et comme dans mon traitement au niveau de l'autre branche j'ai le littéral **-b** alors il suffit de prendre la valeur du littéral du registre de sauvegarde c'est à dire (valeur = echec) et de l'affecter au littéral **-b** rencontré sans faire un autre parcours ou un autre traitement pour ce dernier **(voir arbre ci-dessus).**

Présentation de l'algorithme de PRODUCTION:

Une production est une suite finie de couples **<Pi,Ai>**.Chacun de ces couples
est une étape de production PR dans laquelle **Pi** est une clause , la clause en production et **Ai** un arbre . Cette suite a les propriétés suivantes :

a-La première étape de production PR est :

<P1,A1> = <0,l1,.....(lm)> où
les **lj** sont les littéraux d'une même clause **c** de C l'origine de la production .
Chacun de ces littéraux apparaissant une et une seule fois .la clause en production est vide .

b-La dernière étape de production est :

<Pn,An> si **An** est l'arbre vide , la clause **Pn** est alors la clause produite par la production **PR** .

c-Si la K-ième (k<n) étape de production est :

<P,A> = <P,(LS)B>
où **(LS)** est une branche de feuille **L** (le littéral à éffacer) , **B** un arbre et **P** la clause en production alors l'étape suivante est l'une des formes :

forme N°01: on met en production le littéral **<l>** si l'étape suivante est : **<LP,B>**

forme N°02: on effectue une résolution si l'étape suivante est :
<P,L'1LS)(L'2LS).....,(L'tLS)>
où les littéraux **L'i** sont tels que les conditions I et II sont vérifiées
Condition I :
Il existe une clause **c'** de **c** , la clause appelée a cette étape :
. qui contient l'opposé de l .
. dont aucun littéral n'est dans la branche **(LS)**
(les littéraux de cette branche sont les ascendants ou les A-ancêtres).

. dont aucun littéral n'a pour opposer un littéral de la clause en production P (les littéraux de cette clause sont les littéraux en production).

Les trois dernières conditions sont les conditions de non répétition.

Condition II :

Les L'i sont alors les littéraux de la clause c' qui ne sont pas immédiatement effaçables c'est-à-dire ceux :

- dont l'opposé n'est pas dans la branche **(LS)**.
- qui ne sont pas égaux à une feuille de l'arbre **B**.
- qui ne sont pas égaux à un littéral de la clause en production .

Exemple de Production :

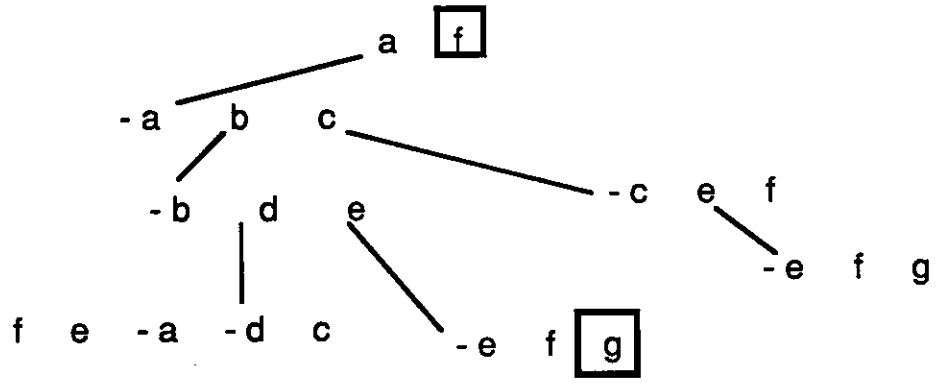
Soit le système de clauses C suivant :

C= { (a,f) , (-a,b,c) , (-b,d,e) , (f,e,-a,-d,c) , (-c,e,f) , (-e,f,g) }

a-Production avec B-ancêtres :

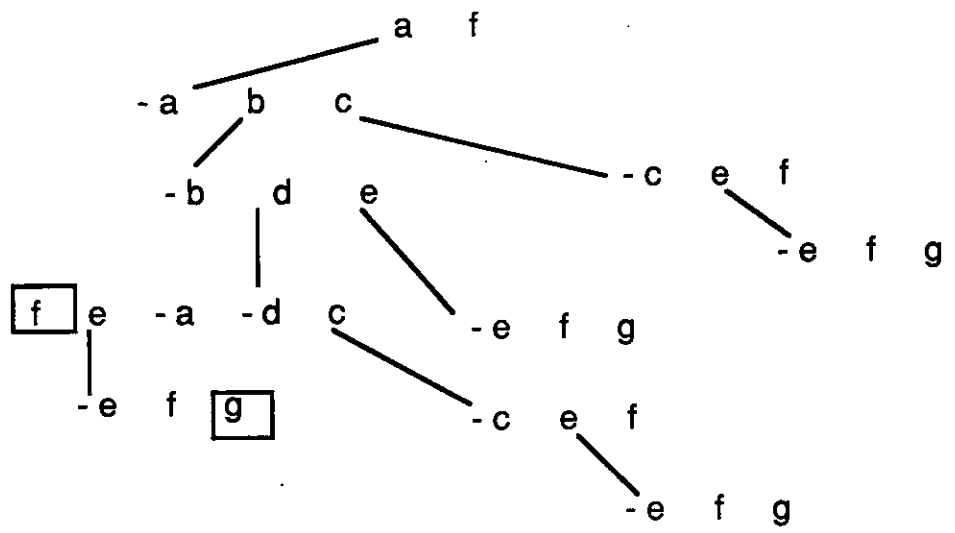
(La clause produite pour cet exemple est : f g .)

Pour cet exemple à partir d'un ensemble de clauses E , on essaye de produire des clauses en partant d'une clause initiale choisie de l'ensemble .
La production consiste à regrouper les littéraux qui n'ont pas d'opposés dans les autres clauses et ce regroupement represente **la clause produite** .



b-Production sans B-ancêtres :

La clause produite pour cet exemple est : f g .



Remarque :

Le Backtracking permettra de déterminer toutes les autres clauses produites

.

Idées principales de l'amélioration :

1- Est-il possible de profiter du registre de sauvegarde que C.CUBADDA et M.D.MOUSSEIGNE ont proposé.
2- Sous qu'elle condition dans l'algorithme de production peut -on utiliser le raffinement .
3- Y-aura t-il perte ou non des clauses produites .

à suivre.

DOUZZAVEDIBISAR

HORREUR !

Quand Basile était tout seul, il adorait faire d'affreuses grimaces en se regardant devant un miroir. Au début, les grimaces n'étaient pas si horribles que ça, bien que Basile tentât de périlleuses acrobaties avec les muscles de son visage. Il avait vu faire des copains qui s'introduisaient deux doigts dans la bouche pour l'élargir pendant qu'un index repoussait le nez vers le haut et que les yeux roulaient dans les orbites. Les diverses variations sur ce thème ne l'avaient pas emballé, c'était plus ridicule qu'affreux. Et puis un peu au hasard, entre deux grands écarts entre mâchoire et oreilles il s'était aperçu que de subtiles déformations pouvaient présenter des traces d'horreur bien plus remarquables.

Basile et son Papa se promenaient à la foire de la Bourse, qui n'avait d'ailleurs plus l'éclat de jadis, et Basile ne se doutait pas que cette fête qui n'en était déja plus guère une, n'allait pas tarder à disparaître pour donner le jour au Centre Bourse. Ce qui amusait surtout Basile c'était la profusion des glaces dans lesquelles, du coin de l'œil il voyait défiler les têtes des gens, et la sienne aussi et celle de son Papa. Les gens font toujours plus ou moins, sans s'en rendre toujours compte diverses moues qui tendent parfois à de véritables rictus. Quel régal pour notre ami Basile qui s'entraînait à les imiter. Il acquit ainsi une véritable virtuosité. Son Père avait bien remarqué le manège et cela avait tendance à le crisper, car son fils prenait de la sorte des têtes impossibles.

Là où les choses se gâtèrent c'est quand ils rencontrèrent la mère Panissette. Basile supportait mal la mère Panissette. Cette brave dame ne savait pas parler sans remuer son visage comme remue un gant quand vous essayez de l'enfiler, en plus, son dentier déficient l'obligeait à agiter sa langue entre ses brêches-dents, d'une manière tellement ostensible, qu'on aurait dit qu'elle débitait ses fadaises en mâchant du jambon. Sans parler de ses épithètes excessives...

« ô ce petit qu'il est beau ..» s'exclama-t-elle au beau milieu de la foule, plantée les bras en croix, ce qui ne l'empêchait pas de continuer à remuer sa langue entre ses chicots. « Fernand tu as le plus beau petit du monde ... » Basile ne savait plus ou se mettre, en même temps il sentait son poil se hérisser, et ce fut malgré lui mais fulgurant il improvisa une horrible grimace. Dire combien elle était insolente est un peu faible, dire combien la mimique de la mère Panissette était finement caricaturée est également faible.

Fernand était coincé entre un fou-rire et la honte du manque de respect envers la vieille. A cette époque on ne badinait pas avec ces choses, les temps ont bien changé. ßasile dit-il sévèrement tu es

malpoli envers Madame Panissette, je ne veux plus te voir faire des grimaces .. Enfin, reprit-il pour contenir son rire me vois-tu, moi, faire des grimaces !

Basile, se dit qu'effectivement ce serait trop horrible et il en fut profondément vexé.

Fernand et Claudine roulaient gaiement sur l'autoroute des vacances, Basile était derrière un coude sur chaque dossier des sièges de ses parents, et son menton posé sur ses mains à plat. Il écoutait ses "vieux" et la route défilait devant ses yeux. Mais dans sa tête il imaginait tout un théatre de grimaces qu'il n'osait plus arborer ... Le paysage lui en inspirait certaines, une colline avec deux maisonnettes pour les yeux et le trait fin et diagonal d'un chemin pour une bouche sarcastique. Une vision un peu floue transformait un énorme pylone électrique en une tête de crapaud un peu avachie.

Fernand aimait bien rouler à cent-dix, c'est bon pour la mécanique, disait-il et on avance vite si on roule régulièrement. Mais c'est une allure impossible à tenir sur l'autoroute à deux voies, à droite on a les quatre-vingt-dix et à gauche on a les cent-quarante et plus. Il faut donc louvoyer en permanence, et Fernand râlait-il en permanence. Aussi, au bout d'un moment le ballot attaché sur la galerie se mit-il à osciller. Arrêt sur la bande d'urgence, avec les bolides qui vous frôlent à moins d'un mètre, Fernand prit deux courroies supplémentaires dans le coffre, et s'en passa une autour du cou pendant qu'il attachait l'autre sur l'arrière de la galerie, bousculé par les rafales de vent des nombreuses voitures; Fernand se mit à nager vers l'avant quand tout-à-coup, frôlé par une deuch qui se faisait doubler, la courroie qui voltigeait autour de son cou fut happée par le tacot et Fernand projeté à terre. Claudine et Basile muets d'horreur se précipitèrent, pendant qu'une voiture s'arrêtait auprès de Fernand étendu. C'est Bruno qui en descendit avec Etiennette sa femme accompagnée de la Mère Panissette, qui se pencha toujours spectaculaire « Oh mon pauvre Fernand qu'est-ce-qu'ils t'ont fait ?...» A cet instant Fernand se retourna vers la mère Panissette, et les yeux révulsés sortit une langue démesurée qu'il accompagna d'une abominable grimace ... l'horreur quoi!

E. B.

<u>La pensée du jour</u>:

Tant qu'on est vivant, c'est pour l'éternité ... à partir du moment où on est mort, c'est comme si on n'avait jamais existé.

Citations:

Plus il y a de contraintes, plus il faut contraindre. C'est la route du chaos. (Aphorisme panspechi)

Quand les moyens de la violence sont répandus partout, rien n'est plus dangereux pour les puissants que de semer la haine et l'injustice, car l'injustice et la haine, à leur tour appelleront d'inévitables représailles. (Manuel du BuSab)

Si vous vous jugez désarmé et inefficace, il ne fait aucun doute que vous allez créer un gouvernement de type despotique pour vous guider. Le despote avisé, par conséquent, a intérêt à entretenir chez ses sujets le sentiment général qu'ils sont désarmés et inefficaces. (Les enseignements de la planète Dosadi, poinrt de vue gowachin)

Peut-on dire que le peuple soit informé et consentant, lorsque la minorité au pouvoir agit en grand secret pour allumer une guerre dans le seul but de justifier l'existence de ses propres forces armées ? ... (Extrait du procès des procès)

Le cycle des saboteurs. DOSADI . Franck Herbert.