Optimisation multicritère de contrôleur flou par une stratégie d'évolution approchant la zone de Pareto

L.Gacôgne * **

* LAFORIA CNRS - Université Paris VI 4 place Jussieu 75252 Paris 5°
tel : 44 27 70 02 fax : 44 27 70 00 mail : gacogne@laforia.ibp.fr

** Institut d'Informatique d'Entreprise (CNAM) 18 allée J.Rostand 91025 Evry

Résumé

Il est, généralement difficile de ramener l'optimisation de plusieurs critères à celle d'un seul critère car sa définition pose problème et de légères modifications sur l'agrégation employée entraîne le processus d'optimisation choisi vers des solutions fort éloignées. Nous proposons donc une stratégie d'évolution adaptée, non pas à la recherche d'une solution, mais d'un ensemble de solutions convergeant vers l'ensemble de Pareto défini pour tous les critères. Nous discutons alors de l'incidence de la taille de la population de solutions et d'un ordre éventuel sur cette population. Nous appliquons cette stratégie sur le problème de l'identification globale (prédicats, règles et divers paramètres) d'un contrôleur flou à deux sorties destiné à guider un robot devant franchir un certain nombre de portes.

Introduction

Dans les problèmes d'optimisation multicritère, la démarche courante consiste à définir une combinaison, linéaire ou non, de tous les critères dans le but d'optimiser cette fonction. Cependant, pour des critères hétérogènes tels que coût, caractéristiques physiques et performances suivant diverses épreuves, il est très difficile de donner une combinaison de ces critères d'autant qu'une faible variation des poids accordés à ces différents critères, entraîne généralement des solutions assez différentes. Par ailleurs, le choix même des épreuves servant à l'estimation globale d'une solution est délicat. Pour un problème concret, établir des seuils, accorder des bonus suivant les performances d'une éventuelle solution, se fait par un jeu d'essais et une bonne solution suivant une telle formule d'agrégation des critères est souvent décevante pour chacun de ces critères, voire mauvaise en modifiant légèrement la formule d'agrégation.

L'idée maîtresse de la recherche de la zone de Pareto, consiste, au lieu de chercher une bonne solution, à obtenir toute une population de solutions, les meilleures possibles suivant tous les critères mais incomparables deux à deux, en laissant le choix à l'utilisateur parmi un petit nombre de ces solutions. Une vision concrète de ce problème a été étudié par [Korhonen, Laakso 85]. Leur première idée était d'implémenter une méthode itérative en orientant la recherche suivant les directives de l'utilisateur. Pour des solutions codées suivant deux dimensions, une représentation graphique de l'ensemble de Pareto est proposée à l'utilisateur [Korhonen, Wallenius 88, 90]. Celui-ci peut changer les poids relatifs à un certain nombre de buts afin de modifier la direction de recherche et sa rapidité.

En ce qui concerne les algorithmes génétiques, [Holland 75] rappelons qu'ils consistent, en partant d'une population d'éventuelles solutions au problème, à coder ces solutions, par exemple comme chaînes de caractères afin que s'exercent sur elles des "opérateurs génétiques". De génération en génération, certaines de ces solutions (des "chromosomes") sont choisies aléatoirement afin d'échanger une partie de leur code (c'est le "cross-over"), et, suivant une faible probabilité, sur certains chromosomes, un opérateur de "mutation" modifie aléatoirement une partie du code. Un grand nombre d'idées peuvent alors être employées pour sélectionner les chromosomes sur lesquels vont s'exercer les opérateurs et pour contrôler l'hétérogénéité de la population. Les algorithmes génétiques, et plus généralement les stratégies d'évolution, constituent donc des heuristiques adaptables à tout problème d'optimisation.

L'idée d'adapter de telles stratégies à une optimisation multicritère pour trouver une population de "bonnes" solutions au lieu d'une solution unique est apparu avec [Schaffer 85] qui considère une partition de la population en relation avec chacun des critères, puis avec [Kursawe 91] et enfin [Horn, Nafpliotis, Goldberg 94] appliquant des algorithmes génétiques à "niches écologiques".

Pour garder le plus longtemps possible, plusieurs minima locaux des fonctions à optimiser, ils établissent une partition de la population (des "niches"), en migrant entre ces niches périodiquement [Franz 72]. A propos d'un problème multicritère en génie chimique, on trouve chez [Viennet Fonteix, Marc 95] une application des algorithmes génétiques, s'exerçant sur chaque critère séparément.

Notre but est de donner une adaptation de stratégie d'évolution pour des problèmes multicritères, avec différents traits originaux, et de l'adapter au problème du réglage de tous les paramètres d'un contrôleur flou dans le guidage d'un robot.

Après avoir défini et donné quelques exemples de la zone de Pareto pour un problème multicritère, la seconde partie de cet article expose, comment en partant d'une faible population aléatoire dans l'espace de recherche, nous conservons toujours les meilleurs individus incomparables au sens de Pareto, en poursuivant sans cesse l'espace de recherche. Nous utilisons pour cela une population de taille variable, à l'intérieur de certaines limites, ainsi qu'une population d'opérateurs génétiques définis en relation avec le problème considéré et évalués suivant le cumul de leur performance à améliorer les individus.

Dans la section III, nous appliquons cette stratégie à la recherche de "bons" contrôleurs flous guidant un robot à travers des portes. Nous définissons une famille de prédicats pouvant recouvrir un grand nombre de celles qui sont couramment utilisées et représentons un contrôleur flou comme code de cette famille de prédicats et une liste variable de règles.

Le problème du passage des portes ou suivi d'une trajectoire matérialisée par une ligne ou une suite de bornes est typiquement un problème d'optimisation multicritère, il est en effet très difficile d'obtenir de bonne solution en cherchant à optimiser une combinaison linéaire, par exemple, de la longueur réalisée, de la distance à la courbe à suivre, du nombre de variations de directions, de variations de vitesse, de l'importance du jeu de règles...

I L'ensemble de Pareto

Soit f_1, f_2, ..., f_p des fonctions de R^n dans R. Le but consistant à trouver des solutions x de R^n minimisant chaque f_i n'est évidemment pas possible dans la plupart des cas, aussi, on définit un ordre partiel dans l'espace de décision R^n par :

$$x < x' \Leftrightarrow \forall i \;\; f_i(x) \leq f_i(x')$$

(On dit que x domine x')

On dit que x et x' sont comparables si $x < x'$ ou $x' < x$, et incomparables dans le cas contraire, un élément premier ê est un élément tel qu'il n'existe aucun e vérifiant $e < ê$. L'ensemble de Pareto associé à cet ordre, est l'ensemble de tous les éléments premiers de R^n (sans prédécésseur) [Sawaragi, Nakayama, Tanino 85].

Les exemples suivants sont donnés pour des "chromosomes" qui sont tous les couples (x, y) de $[-1, 1]^2$. Donnons d'abord une illustration de l'ensemble de Pareto pour 4 fonctions de R^2. La figure 1 montre une partie de l'ensemble de Pareto (ici un carré) pour les quatre critères à minimiser

$f_1(x) = |x + y - 1|$, $f_2(x) = |x - y - 1|$,
$f_3(x) = |x + y + 1|$ et $f_4(x) = |x - y + 1|$.

Il est bien sûr facile de trouver le bord de l'ensemble de Pareto, en triant l'ensemble suivant $f_{min} - f_{max}$, et de trouver les 4 sommets, en la triant suivant la plus grande somme des variations des 4 critères, ce score étant (0 2 2 0) (0 0 2 2) (2 0 0 2) (2 2 0 0) pour (f_1, f_2, f_3, f_4) aux sommets, (0 1 2 1) ... aux milieux des côtés, et (1 1 1 1) au centre. Cependant aucun de ces procédés ne peut être généralisé pour trouver le contour d'un ensemble de Pareto.

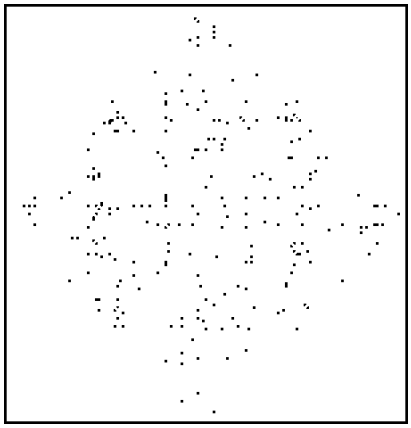

Figure 1 Population de 250 chromosomes obtenue en 50 générations sans aucun ordre pour les 4 fonctions à minimiser |x + y - 1|, |x - y - 1|, |x + y + 1|, |x - y + 1|.

L'ensemble de Pareto peut recevoir des formes inattendues, par exemple non connexe :

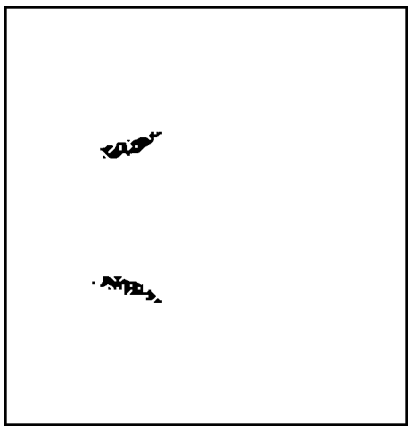

Figure 2 La 50^e génération d'une population de 250 points qui converge vers les deux points de Pareto (-1/2, ±1/2), pour les deux critères $f_1(x, y) = (2x + 1)^2$ et $f_2(x, y) = 1 + |\cos \pi y(1 - x)|$

La frontière elle-même peut ne pas être connexe (couronne de la figure 3).

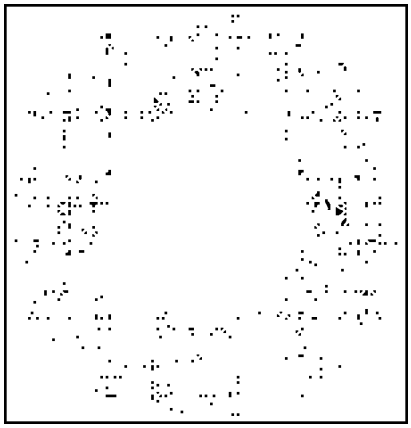

Figure 3 La 50e population de 500 chromosomes sans ordre, pour $100|x^2 + y^2 - 1|$ et $100|x^2 + y^2 - 1/4|$

L'ensemble de Pareto est non convexe dans la figure 4 pour deux fonctions, (exemple du à [Viennet, Fonteix, Marc 95]).

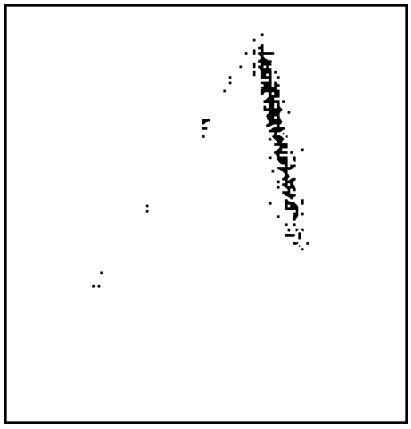

Figure 4 Une population de 200 points obtenue en 50 générations pour deux critères

$f_1(x, y) = 200(1 + 3x - 2y)^2 + 4(1 + 4x - 4y)^2$ et

$f_2(x, y) = 200(1 - 2x)^2 + 7(1 + 4y)^2$ dont les minima sont respectivement (-1/2, -1/4) et (1/2, -1/4).

II Une strategie d'évolution pour l'optimisation multicritère

II-1° Taille de la population et élimination

Le but des algorithmes génétiques est la recherche d'une solution acceptable à un problème numérique, c'est à dire pour une seule fonction à valeurs dans R. Nous souhaitons, maintenant trouver un nombre (le plus modeste possible) de solutions acceptables en regard de plusieurs critères numériques, de telle sorte que l'utilisateur puisse choisir au sein de cette population.

Nous partirons toujours d'une population aléatoire P formée par quelques individus (P_{min} chromosomes). Nous fixons cette taille minimale P_{min} afin de poursuivre l'exploration de l'espace de recherche au cas où la taille de la population descend en dessous de ce seuil, et nous fixons de même pour des raisons de calcul, une taille maximale P_{max}. Au commencement de chaque session, nous éliminons de la population initiale P_0 les plus mauvais éléments s'il y a des comparaisons possibles et dans ce cas, nous cherchons d'autres chromosomes jusqu'à ce que $|P_0| = P_{min}$. Un opérateur génétique spécifique appelé "migration" est utilisé pour cela.

A la génération t la population P_t contient un nombre de chromosomes entre ces deux bornes P_{min} et P_{max}, et tous les éléments de P_t sont incomparables deux à deux, nous appliquons alors sur P_t tout entier des opérateurs génétiques.

Lorsqu'un opérateur génétique est appliqué sur x, en donnant x', nous comparons x et x', et s'ils sont comparables, nous supprimons le plus mauvais (évolution lamarckienne). Au cas où x' est le meilleur ou bien incomparable avec x, nous devons le comparer avec tous les autres individus de la population de façon à ne conserver que des individus incomparables deux à deux.

Lorsque le croisement est appliqué, l'élimination s'exerce de la même façon entre les deux parents et les deux enfants, et le cas échéant entre les deux enfants et le reste de la population.

De cette manière, la taille de la population est généralement croissante dans une première phase et décroissante par la suite.

A la fin de chaque génération, il est possible que $|P_{t+1}| > P_{max}$, il est alors nécessaire de supprimer des chromosomes, et plusieurs options sont alors possibles :

Si nous trions la population suivant une formule d'agrégation comme la moyenne des fonctions $f_1, f_2, ..., f_p$, nous revenons au problème de l'optimisation d'une seule fonction avec tous les défauts que nous avons indiqué.

Si, par contre, nous demeurons avec le "score" d'un chromosome x défini comme le p-uplet $(f_1(x), f_2(x), ..., f_p(x))$ nous pouvons trier la population suivant l'ordre lexicographique. Avec cet ordre total, on a par exemple pour 4 critères :

$(1\ 1\ 1\ 2) < (1\ 2) < (2\ 1\ 0) < (2\ 2\ 2) < (2\ 2\ 3)$.

En faisant cela nous donnons une prépondérance aux premiers critères de la liste (f_1, f_2, ..., f_p), et, pour un problème donné, l'utilisateur n'a donc qu'à indiquer l'ordre d'importance des critères à optimiser.

Une autre solution plus longue, est de proposer l'union des populations obtenues par les p! permutations des p critères.

Exemple

Si nous regardons 3 fonctions, nous pouvons réaliser 6 recherches suivant l'ordre que nous donnons à ces 3 fonctions, afin d'avoir une idée de l'ensemble de Pareto. Par exemple, avec f_1, f_2, f_3, nous déroulons 50 générations sur chacune des 6 populations respectivement triées par : (f_1, f_2, f_3), (f_1, f_3, f_2), (f_2, f_1, f_3), (f_2, f_3, f_1), (f_3, f_1, f_2), (f_3, f_2, f_1). Ces 3 fonctions sont les cônes :

$f_1(x, y) = 50(x - 0.5)^2 + 80y^2$,

$f_2(x, y) = 100(x + 0.5)^2 + 50(y + 0.5)^2$ et

$f_3(x, y) = 50(x + 0.5)^2 + 100(y - 0.5)^2$,

Leurs minima 0 sont respectivement en (0.5, 0), (-0.5, -0.5) et (-0.5, 0.5).

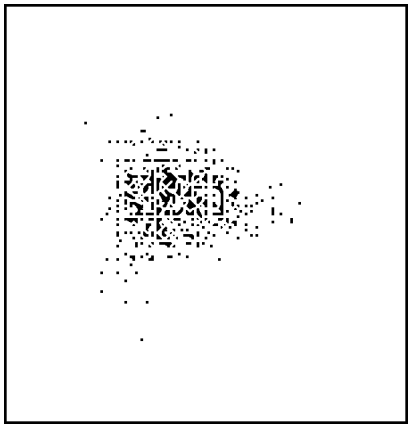

Figure 5 La 50e génération d'une population de taille 300 pour les 3 critères sans aucun ordre.

On peut voir après 50 générations un déplacement de toute la population dans R^2 vers les trois sommets quand nous faisons le tri avec f_1, f_2 ou f_3 en premier.

Figure 6 Superposition de 7 populations (chacune est la 50e génération avec 200 chromosomes) pour les 6 ordres lexicographiques. Le nuage central est obtenu par tri de la population avec la seule moyenne $(f_1 + f_2 + f_3) / 3$.

Remarque Supposons les p critères X_1, X_2, ..., X_p stochastiquement indépendants et uniformément répartis dans l'espace de recherche. La probabilité de les trouver dans un ordre donné est $pr(X_1 < X_2 < ... < X_p) = 1 / p!$

Et, en posant $I = (X_1, X_2, ..., X_p)$, on obtient $pr[I \text{ et } I' \text{ comparables}] = 2pr\ (X_1 < X'_1 \text{ et } X_2 < X'_2 \text{ et ... et } X_p < X'_p] = 1 / 2^{p-1}$, et si $P = \{I_1, I_2, ..., I_n\}$ est une population de taille n dans l'espace de recherche, on a :

$$pr(I_1,\ldots,I_n \text{ incomparables 2 à 2}) = \left(1 - \frac{1}{2^{p-1}}\right)^{\frac{n(n-1)}{2}}$$

Donc, pour une même probabilité pr, d'avoir une population d'éléments deux à deux incomparables vis à vis de p critères, il faut que cette population de taille n soit telle que :

$$n(n-1)\log(1 - 1/2^{p-1}) = 2\log(pr)$$

Dès que $p > 2$, on peut faire l'approximation $n(n-1) = - 2^p \log(pr)$, ce qui signifie que la taille n doit être de l'ordre de $2^{p/2}$. Dans la pratique, cela permet de fixer P_{min} assez bas pour favoriser les migrations par exemple, de l'ordre de 10 pour 2 critères, 15 pour 3, 20 pour 4 et 150 pour 10. Nous avons donc fixé P_{max} de l'ordre de $5*2^{p/2}$.

II-2° Contrôle de la population des opérateurs

Nous présentons maintenant tous les détails relatifs à l'implémentation de notre stratégie d'évolution.

a) Une population aléatoire P_0 est produite avec la taille P_{min} ainsi qu'une population OP_0 (de taille > P_{min}) de différents opérateurs.

Ceux-ci sont de plusieurs type (mutation, transpositions de deux gènes, bruits gaussiens, création ou suppression, ou changement de priorité dans une règle floue aléatoirement choisie comme on le verra plus loin, ...) qui sont initialement produits aléatoirement dans la population OP_0.

Un chromosome est une liste de longueur variable de règles floues (section III). En introduisant de plus, la force d'une règle à l'intérieur d'un chromosome comme la moyenne (ramenée dans [0, 100] de son utilisation au cours d'une épreuve), nous observons le rôle joué par des opérateurs génétiques tels que la suppression de la règle la moins utile, ou une mutation exercée sur la règle la plus utilisée.

Remarque Lors d'expériences précédentes [Gacôgne 94] nous avions des chromosomes de longueur fixe codant des tables de règles pour un contrôleur flou à deux entrées et une sortie et une famille de prédicats figée. Les opérateurs génétiques étaient alors une famille de fonctions mathématiques (la mutation modifiant le

cinquième site avec le gène "NB" par exemple) dont seul l'ordre pouvait diriger l'aspect aléatoire de la recherche. Ici, nous nous intéressons au type d'opérateurs, afin d'observer leur influence au cours de longues évolutions.

b) Dans tous les cas, le processus de passage d'une génération à la suivante s'écarte considérablement des modes de sélection des algorithmes génétiques standards. Nous appliquons, en effet, le premier opérateur de OP_t au premier individu de P_t, (donc le meilleur opérateur avec le meilleur individu) puis le second avec le second etc. En cas de croisement le second parent est choisi aléatoirement parmi les prédécesseurs ou les suivants du premier parent, suivant le plus grand de ces deux ensembles. Nous n'avons donc aucun paramètre de probabilité à fixer.

Lorsqu'un opérateur est appliqué, le tri est immédiatement réalisé entre parents et enfants pour ne conserver que les meilleurs individus incomparables deux à deux. De cette façon, lorsque s'amorce une descente vers un minimum local d'un des critères, seul le meilleur individu relatif à ce minimum est conservé.

c) Si le but de notre algorithme est de trouver des minima au sens de Pareto pour une liste de fonctions évaluées sur toute une population de solutions potentielles à un problème donné, nous évaluons en même temps les opérateurs génétiques en les triant à chaque génération.

Nous attribuons pour cela à chaque opérateur, sa capacité algébrique à diminuer les fonctions $(f_1, ..., f_p)$ à optimiser. La procédure est simplement constituée par une initialisation à 0 et une incrémentation de $\sum f_i(x') - f_i(x)$ chaque fois que l'opérateur est appliqué à x en donnant x'. Cette "performance" de l'opérateur est réinitialisée au cas où aucune amélioration n'est observée au cours d'une génération. Ceci a pour effet d'éviter que les opérateurs ne convergent vers une population stable ou périodique. Si par exemple une grande importance est accordée au critère de longueur pour un chromosome, l'opérateur de "suppression" sera bien noté.

d) Contrôle des opérateurs : si au moins un progrès est réalisé lors d'une génération, nous trions les opérateurs et créons un nouvel opérateur du même type que celui du meilleur. Nous avons ainsi une population fluctuante d'opérateurs qui permet à l'utilisateur de définir une grande variété d'opérateurs dont il pense qu'ils peuvent être adaptés à son problème. Notre algorithme permet alors de faire évoluer cette population en éliminant les opérateurs les moins performants.

III Application à la mise au point d'un contrôleur flou

Nous appliquons l'algorithme d'évolution décrit plus haut à la recherche de contrôleurs flous pouvant guider un véhicule autonome. Le problème n'est pas difficile si on se fixe un objectif limité, mais il devient singulièrement complexe lorsqu'on souhaite trouver non seulement des bonnes règles, mais aussi une famille de prédicat, les univers des différents paramètres et même les grandeurs d'entrée. Ainsi pour le suivi d'une ligne, il n'y a pas de réponse simple à la question des capteurs (quelles sont les bonnes distances à mesurer ?)

III-1° Contrôleur $[-1, 1]^n \rightarrow [-1, 1]^p$

Si A est un ensemble flou défini sur l'univers U, nous notons μ_A la fonction d'appartenance de A, et $supp(A) = \{x \in A / \mu_A(x) \neq 0\}$ le support de A ainsi que $ker(A) = \{x \in A / \mu_A(x) = 1\}$ le noyau de A. Nous définissons des contrôleurs flous de $[-1, 1]^n$ vers $[-1, 1]^p$:

Règle

Nous attribuons des ordres de priorité pour séparer les règles en générales et particulières [Gacôgne 93]. Si A est un ensemble flou de fonction d'appartenance μ_A continue sur $[-1, 1]^n$, k est un entier et $c \in [-1, 1]^p$, nous disons que le triplet (k A c) est une règle de priorité k, c'est une formalisation de "si (x est A) alors (y est c) sauf si une règle plus prioritaire (d'ordre inférieur à k) s'applique".

Contrôleur flou avec défauts

Si $k \in N$ et C_k est un ensemble fini $\{(k\ A_1\ c_1) ... (k\ A_i\ c_i) ...\}$ de règles de même priorité avec $c_i \in [-1, 1]^p$, la fonction de Sugeno f_k [Sugeno 85] définie sur la réunion des supports des A_i est habituellement pour $x \in [-1, 1]^n$:

$$f_k(x) = (\sum \mu_{Ai}(x) c_i) / (\sum \mu_{Ai}(x)).$$

Un contrôleur flou avec défauts C est, pour nous, un ensemble de règles dont les ordres de priorités sont tous les entiers de 0 à m (0 pour les règles les plus particulières, et m pour les plus générales). La fonction de Sugeno associée au contrôleur sera définie pour tout $x \in [-1, 1]^n$ par $f(x) = f_k(x)$ si k est le premier ordre tel que x soit dans le domaine d'une f_k, et 0 sinon.

(Cette dernière condition est équivalente à une dernière règle plus générale que toutes les autres et dont la conclusion serait 0 sans condition.)

On dit que 0 est l'ordre des règles les plus spécifiques, et m, de l'ordre de 0 à 3, celui des règles les plus générales. Nous supposerons C minimal, (il n'est pas possible d'obtenir la même fonction avec un sous ensemble propre de C), mais bien sûr de tels ensemble minimaux ne sont pas uniques.

Remarque Il est possible d'imaginer un système où chaque f_k peut être étendu à [-1, 1] en convenant 0/0 = 0 et où $f(x) = \phi(f_0(x), ..., f_n(x))$.

ϕ pouvant être $\phi(a_1, ...a_n) = (\sum m_i a_i) / \sum m_i$, où m est décroissante, par exemple, $m_i = n + 1 - i$. Voir [Yager 92] pour un système analogue.

III-2° Familles de prédicats utilisées

Nous souhaitons appliquer notre stratégie d'évolution pour la mise au point de contrôleurs flous utilisant des prédicats relativement variables mais définis par un petit nombre de paramètres. Ceci a déjà fait l'objet de nombreux travaux liés aux algorithmes génétiques, cependant, il est assez difficile de considérer le problème dans son entière généralité. La plupart du temps de fortes restrictions permettent d'obtenir des résultats acceptables mais la recherche ne s'appliquant que sur un aspect du problème (les règles ou une famille de prédicats bien définie). Nous avons cherché ici à réaliser un compromis entre une famille générale et un petit nombre de paramètres. Le quadruplet (nb, contr, trap, fuz) défini ci-dessous permet d'obtenir des familles très diverses recoupant un grand nombre de familles de prédicats utilisées couramment dans les applications concrètes.

Nous définissons ces familles à partir de fonctions d'appartenance sur [-1, 1] grâce à :

Le nombre $p \in \{1, 2, 3, 4\}$ (figures 7, 8 et 9) qui permet de déduire 2p + 1 prédicats.

Le second $r \in [0, 1]$ sera nommé "contraction" des prédicats, il mesure la manière dont sont resserés ces prédicats autour de 0.

Le coefficient $h \in [0, 1]$ est un indice ("trap") de chevauchement, et $c \in]0, 1]$ ("fuz") est une mesure du flou grâce à la "pente" des trapèzes.

Soit m_k ($0 \le k \le p$ avec $m_{-1} = m_1$) le "sommet" d'une fonction triangulaire definie par :

$$m_k = r\left(\frac{k}{p}\right)^2 + (1 - r)\frac{k}{p} \quad \text{et :}$$

$$\mu_k(x) = \min\left(1, \max\left(0, h + 1 - \frac{1}{c}\left|\frac{m_k - x}{m_k - m_{k-1}}\right|\right)\right)$$

Chaque fonction d'appartenance est la troncature d'une "fonction triangulaire" dont le maximum est $1 + h + r.m_k$ en m_k, (ainsi $m_0 = 0$) et qui est linéaire entre ce sommet et le point (m_{k-1}, 0), symétrique autour de m_k et tronquée dans [0, 1].

On nomme ZE l'ensemble flou de fonction d'appartenance μ_0 et on impose une symétrie telle que $\mu_{-k}(x) = \mu_k(-x)$. On peut alors appeler les prédicats PS et NS pour k = 1 ou -1 dont les fonctions sont respectivement μ_1 and μ_{-1}, PM and NM pour μ_2 and μ_{-2} et PB, NB pour ± 3.

Les figures suivantes montrent des possibilités de familles sur [0, 1].

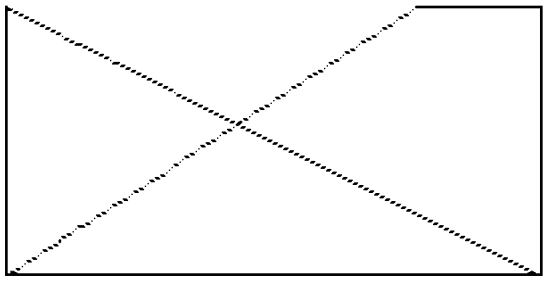

Figure 7 Exemple de famille de predicats sur [0, 1] avec nb = 1, contr = 0.3, trap = 0 et fuz = 1.

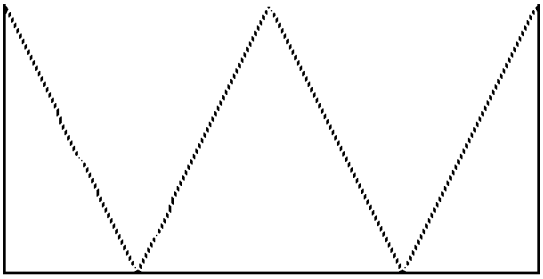

Figure 8 nb = 2, contr = 0, trap = 0 et fuz = 0.5.

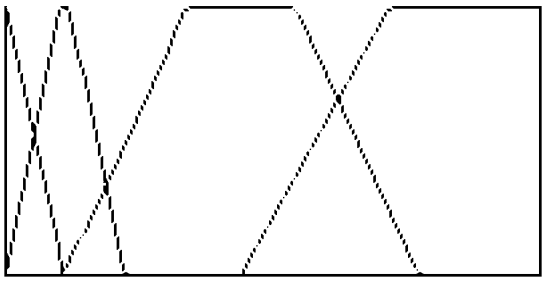

Figure 9 nb = 3, contr = 1, trap = 0 et fuz = 1.

II-3° Application à la simulation d'un véhicule autonome

Codage d'un chromosome

Les chromosomes n'ont pas une longeur fixe, leur structure est (nb contr trap fuz ouv R_1 R_2 R_3 ...) où nb, contr, trap, fuz sont les coefficients décrits plus haut, "ouv" un angle en degrés, et R_i les règles, toutes de la forme (pr f h_1 ... h_{nh} c_1 c_2 ... c_{nc}).

Les hypothèses h_i sont des symboles parmi {any, nb, ns, ze, ps, pb} et les conclusions c_j

sont des entiers [-100, 100], pr est la priorité de la règle, et f est sa force (III-2-a).

Epreuve réalisée

Il s'agit de faire suivre un parcours matérialisé par des portes, le robot M doit effectuer un slalom en mesurant à chaque étape, les angles qui séparent sa direction des visées des deux côtés P_1, P_2 de la prochaine porte, ainsi que la distance d qui le sépare du milieu de cette porte. Les deux entrées seront constituées par a_1+a_2 et d.

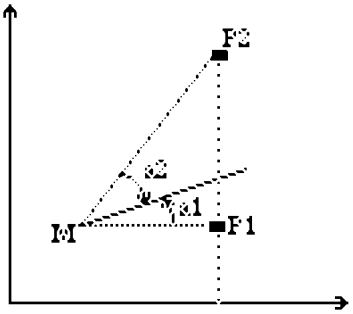

Figure 10 Le mobile M mesure les deux angles et sa distance au centre de la porte à passer.

Plus précisément, nous allons chercher des contrôleurs flous de [-1, 1]*[0, 1] vers $[-1, 1]^2$. Une règle ayant deux prémisses pour a_1+a_2 apprécié dans [-ouv, ouv], et d dans [0, dm] avec dm = 40 et deux conclusions dans [-100, 100] donnant les pourcentages de variations de direction (le maximum est am = 60°) et de vitesse (maximum acc = 10 pixels) à effectuer. Le pas variable restant dans l'intervalle [4, 35].

Critères à optimiser

Nous allons orienter notre recherche avec les quatre critères suivant à minimiser :

1 - Le nombre "np" de portes restantes à l'issue de 35 pas du robot.

2 - Le second critère est un malus attribué aux portes non franchies, c'est la moyenne "val" des distances mesurées au passage de chaque porte si celles-ci sont passées à l'extérieur.

3 - Un autre critère est la "tenue de route" [Gacôgne 94]. C'est le nombre en pourcentage :

$$tr = \left| \frac{|da|}{am} + \frac{dp}{2acc} - \frac{1}{2} \right| \in [0, 1]$$

Dans lequel da et dp sont les changements de direction et de pas réalisés à chaque prise de données, "am" et "acc" étant les maxima de ces changements. Plus tr est proche de 0, meilleure nous estimons la trajectoire (accélération en ligne droite, freinage dans les virages), lorsqu'il est proche 50, nous l'estimons mauvaise, et plus tr est proche de 100, plus la trajectoire est à l'opposé d'une bonne conduite.

4 - Nous tenons enfin compte du nombre "ns" de symboles significatifs dans une liste de règles, afin de favoriser les petits nombres de règles.

Expérience d'évolution

Afin d'explorer continuellement l'espace des solutions, nous avons choisi une taille minimale de 5 individus. Ainsi avec 2 ou 3 critères, il est fréquent que l'élimination des solutions comparables les plus mauvaises amène à 3 solutions retenues à l'issue d'une génération. Pour les 4 critères mentionnés la taille de la population oscille entre une et deux dizaines, c'est pourquoi, nous avons choisi une taille maximale de 25.

Nous avons fait 20 évolutions, chaque expérience étant limitée à 5000 évaluations (5000 parcours du slalom). Ces expériences ont été faites pour moitié en triant la population suivant l'ordre lexicographique sur (np val tr ns), et pour moitié sur (val np tr ns) attendu que les deux premiers critères sont manifestement les plus importants.

A l'issue de ces évolutions nous observons la variété des solutions trouvées et faisons concourir ensemble les meilleurs individus.

Conclusions de l'expérience

L'observation des opérateurs lors des différentes sessions montre l'importance de l'opérateur de "migration" qui permet, surtout en début d'évolution de ne pas converger trop rapidement vers une solution locale. Il est néanmoins évident que ce sont des évolutions séparées suivies de confrontations qui permettent d'obtenir de bons résultats. Ce fait a été remarqué par de nombreux chercheurs qui ont cherché à simuler des "niches écologiques" [Goldberg, Richardson 87]. L'action du "cross-over" doit être freiné en début de parcours par un grand nombre d'autres opérateurs, et on remarque en effet que le croisement n'opère bien qu'en fin de parcours. Nous avons dressé le nombre total de fois où un opérateur est arrivé en tête lors d'une génération, la liste obtenue est : migration (le meilleur opérateur), suppression de la règle la moins utilisée, mutation sur la règle la plus utilisée, crossover, bruit sur une conclusion de règle, mutation générale, suppression d'une règle, bruit sur un prédicat, création d'une règle aléatoire supplémentaire, changement de priorité d'une règle, et bruit sur le paramètre "ouv" qui est donc l'opérateur le moins susceptible d'apporter des améliorations. On pourra cependant remarquer que si on regroupe les opérateurs les plus perturbants (le plus liés à une exploration aléatoire uniforme), ceux-ci sont largement classés en tête.

Remarque En ce qui concerne les individus trouvés, presque tous réalisent des contrôleurs flous basés sur des familles de 5 prédicats (un seul est à 3 prédicats, 4 le sont des avec 7 prédicats) se chevauchant fortement (contr = 0.43

et trap = 0.45 en moyenne) et avec des trapèzes peu spécifiques (fuz = 0.77 en moyenne) c'est à dire assez larges. Ces contrôleurs ont peu de règles (généralement deux et leurs symétriques la plupart du temps ayant le même niveau de priorité) et l'univers d'appréciation des angles est donné par "ouv" allant de 25° à 94° (moyenne 60°).

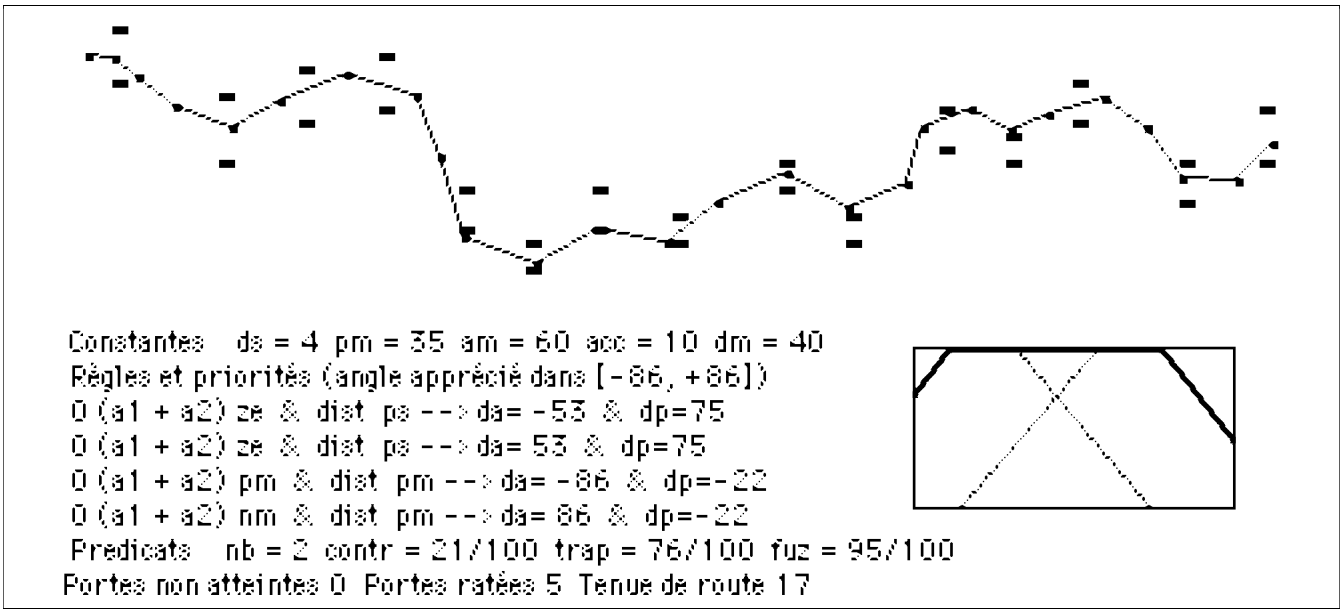

Figure 11 Un des meilleurs résultats (3 portes ratées de peu). Les prédicats NM, NS, ZE, PS, PM signifient respectivement "negative medium" et "small", zero, "positive small" et "medium". Les règles ont même niveau de priorité.

Références

Bäck Th. Evolutionary algorithms in theory and practice. Oxford University Press, New York 1995

Bäck T. Kursawe F. Evolutionary algorithms for fuzzy logic IPMU p.659-664, 1994

Bäck Th. Schwefel H.P. An overview of evolutionary algo. for parameter optimization. Evol. comput. 1 p.1-23, 1993

De Jong K.A. Genetic algorithms are not function optimizers. Foundations of genetic algorithms 2, p.5-17. Morgan Kaufmann Pub. San Mateo, 1993

Franz Non-linearities in genetic adaptative search, Doctoral dissertation, Abstracts International 33 (11) 5240B-5241B University of Michigan, 1972

Gacôgne L. Une stratégie de hiérarchisation des règles dans les systèmes à base de connaissance, Actes des journées Volcan IA p.226-300, 1993

Gacôgne L. About the fitness of fuzzy controllers whose rules are learned by genetic algorithms. EUFIT p.1523-1531, 94

Goldberg D.E. Richardson J. Genetic algorithms with sharing for multimodal function optimization, GA and their applications p.41-49 Hillsdale New Jersey, Lawrence Erlbaum ass., 1987

Holland J.H Adaptation in natural and artificial system. Ann Arbor University of Michigan Press 1975

Korhonen Laakso A visual interactive method for solving multiple criteria problem, European journal of Operational Research Society, vol.32 n°7 p.577-585, 1985

Korhonen P. Wallenius J. A Pareto race, Naval Research Logistics Wiley, vol 35 p.615-623, 1988

Korhonen P. Wallenius J. A multiple objective linear programming decision support system, Decision Support System North Holland vol 6 p.243-251, 1990

Kouada I. Sur la propriété de domination et l'existence de points Pareto-efficaces en optimisation vectorielle convexe, RAIRO Op.research vol 28 n°1 p.77-84, 1994

Kursawe F. A variant of evolution strategies for vector optimization, Proc. PPSN I p.193-197 Sringer Verlag 1991

Rudolph G. Convergence analysis of canonical genetic algoritms. IEEE Transactions on neural networks, Special issue on evolutionary computation, p.63-66 IEEE Press, 1994

Sawaragi Y. Nakayama H.Tanino T. Theory of multiobjective optimization, Academic Press, 1985

Schaffer J.D. Multiple objective optimization with vector evaluated genetic algorithms, Proceedings of an International Conf. on GA and their applications p.93-100, 1985

Schwefel H.P. Numerical Optimization of computer models, J.Wiley 1981

Schwefel H.P. Evolution and optimum seeking. Sixth generation computer technology series. Wiley, 1995

Steuer R.E. Multiple criteria optimization, theory, computation and applications, Wiley 1986

Sugeno M. An introductory survey of fuzzy control. Information and Science n°36 p.59, 1985

Viennet R. Fonteix C. Marc L. Optimisation multicritère à l'aide d'un algorithme génétique diploïde, Evolution Artificielle, Brest, 1995

Yager R.R. Hierarchical representation of fuzzy if-then rules, Proc. IPMU Conference Mallorca, 1992