Apprentissage génétique d'une extension floue du dilemme itéré des prisonniers

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
tel : 69 36 73 10 fax : 69 36 73 05

Résumé

Nous étendons l'approche classique du problème itéré des prisonniers à une notion de trahison-coopération exprimée de façon symbolique ou continue par un degré de confiance. Pour cette généralisation, nous comparons le mode d'inférence habituel dans le style système-expert, à deux autres systèmes d'inférence inspiré des méthodes de contrôle flou de Sugeno et Mamdani. Avec une représentation limitée des stratégies, nous cherchons à optimiser le coût lors de confrontations croisées dans une population grâce aux algorithmes génétiques.

Abstract

We extend the classical approach of iterated prisoners problem towards a fuzzy confidence value of trahison-cooperation in a symbolic or numeric way. For that, we compare the usual inference with a complete set of crisp rules, with two other inference modes based on Sugeno or Mamdani method for fuzzy control. According to a simple representation of strategies, we attempt to reach the cost optimization when confronting the strategies of a population each other, with the help of genetic algorithms.

I Le problème [Axelrod 87]

Deux suspects en détention provisoire sont placés, sans pouvoir communiquer, dans le choix suivant : si l'un avoue mais pas son partenaire, il sera libéré (sa charge sera T = 0, T comme trahir) et l'autre aura un coût D = 5 ans de prison (D = duper), si les deux avouent, ils auront chacun C = 2 ans de prison (C = coopérer), et si aucun n'avoue, ils auront P = 4 ans de prison (P = punir).

Le dilemme réside dans l'hypothèse où l'adversaire avoue, à avouer aussi pour n'avoir que 2 ans au lieu de 5, et à avouer également dans le cas où l'autre n'avoue pas, pour être libéré. Cependant en se taisant tous deux ils n'obtiendraient que 2 ans au bénéfice du doute.

Plus généralement deux partenaires (concurrents, rivaux, ennemis ...) réitèrent un "marché" suivant ces règles et on aimerait trouver une stratégie minimisant en moyenne le coût au cours des itérations.

Le "gain" moyen (en années de prisons pour les prisonniers) est la fonction que l'on va chercher à minimiser. Le fait que (T+D) / 2 > C indique que les deux entités n'aient pas intérêt à s'entendre à tour de rôle pour une série de trahir-duper, la coopération leur étant plus avantageuse.

On gardera toujours T = 0 < C = 2 < R = 3 < P = 4 < D = 5.

R est le renoncement introduit dans [Delahaye 92] que, dans un deuxième temps, on pourra utiliser dans des règles telles que (t c) $\rightarrow$ r, c'est à dire "il a trahi et j'ai coopéré, alors je renonce à continuer", la valeur de R est le coût pour les deux adversaires lorsque l'un d'entre eux abandonne. Sa valeur est choisie de manière à ne pas encourager le renonçement, il y a avantage à trahir ou à coopérer avec un adversaire coopérant. Cette valeur est définitive pour le reste de la partie, le nombre de coups d'une partie restant donc le même.

Nous serons obligé de nous écarter de cette position en définissant le renonçement comme "passer un tour", ce qui accorde R à chacun des deux joueurs, mais non pour la suite de la partie. En effet, lorsque nous étendons la décision à [-1, 1] (-1 étant la trahison, 0 le renonçement et 1 la coopération), le gain devra être calculé par interpolation, et l'esprit même de cette extension à toutes les nuances impose des décisions telles que 0.2 signifiant un abandon non définitif tirant du côté de la coopération par exemple.

Les stratégies

Lorsque ce dilemme est répété plusieurs fois avec les mêmes adversaires, on dira qu'on a une "partie" résultant de (par exemple 10) "coups". Plusieurs stratégies peuvent être alors imaginées :

1) Gentille : on coopère toujours.
2) Méchante : on trahit toujours.
3) Lunatique : C ou T est tiré au hasard.
4) Donnant-donnant : on coopère la première fois, puis on joue ce que l'adversaire a joué au coup précédent.
5) Rancunier : si l'adversaire a trahit au moins une fois dans les coups précédent, alors on trahira toujours, sinon on coopère.
6) Périodique méchant : quelquesoit le jeu de l'adversaire, on coopère une fois puis on trahit deux fois.
7) Périodique gentil : idem mais une trahison suivie de deux coopérations.
8) Majorité-mou : on joue le jeu de l'adversaire qui a été le plus présent au cours des jeux précédents, et la coopération en cas d'égalité.
9) Méfiant : trahison au début, puis donnant-donnant
10) Majorité-dur : idem majorité-mou sauf trahison en cas d'égalité.
11) Sondeur : trahison suivie de deux coopérations pour les trois premiers coups, puis si l'adversaire a coopéré aux coups 2 et 3 alors on trahit toujours, sinon donnant-donnant.
12) Donnant-donnant dur : coopération sauf si l'adversaire a trahit lors de l'un des deux coups précédent.

Problèmes de représentation

En ce qui concerne le problème exposé ci-dessus, une partie peut être représentée par un couple (jeu de l'adversaire, mon jeu) et une itération par une liste de coups. C'est alors qu'une stratégie est, d'une manière générale, une fonction booléenne (à valeurs dans {T, R, C}) de cette liste.

Cependant, dans le but d'engendrer automatiquement de telles stratégies, il est plus aisé de les représenter de façon déclarative comme liste de règles,

(A1 M1) (A2 M2) (A3 M3) ... (An Mn) $\rightarrow$ Décision $\in$ {T, R, C} où A désigne l'adversaire et l'indice est le rang des derniers coups joués à partir du précédent en remontant dans le temps vers le n-ième (non fixe) coup antérieur, ainsi :

"donnant-donnant" sera constitué des trois règles $\varnothing \rightarrow C$, $(T\ _) \rightarrow T$, $(C\ _) \rightarrow C$,

"méfiant" sera $\varnothing \rightarrow T$, $(C\ _) \rightarrow C$, $(T\ _) \rightarrow T$

"gentille" sera $\varnothing \rightarrow C$, $(\ _\ _\ ) \rightarrow C$

"donnant donnant dur" serait déterminé par les règles $(T\ _)\ (T\ _) \rightarrow T$, $(T\ _)\ (C_) \rightarrow T$, $(C\ _)\ (T\ _) \rightarrow T$, $(C\ _)\ (C\ _) \rightarrow C$, $(\ _\ _\ ) \rightarrow C$, $\varnothing \rightarrow C$

"méchant périodique" serait $(\ _\ C)\ (\ _\ T) \rightarrow T$, $(\ _\ T)\ (\ _\ C) \rightarrow T$, $(\ _\ T)\ (\ _\ T) \rightarrow C$

Naturellement la stratégie lunatique, règle du type $(\ _\ _\ ) \rightarrow$ random {C, T}, ne peut se représenter ainsi, et par ailleurs la stratégie "majorité" nécessite un trop grand nombre de règles.

On parlera cependant d'une stratégie "majorité mou" comme celle qui sonde sur 3 coups grâce aux règles :

$(T\ _)\ (T\ _)\ (T\ _) \rightarrow T$,	$(T\ _)\ (T\ _)\ (C_) \rightarrow T$,	$(T\ _)\ (C\ _)\ (T\ _) \rightarrow T$,
$(C\ _)\ (T\ _)\ (T\ _) \rightarrow T$,	$(T\ _)\ (C\ _)\ (C\ _) \rightarrow C$,	$(C\ _)\ (T\ _)\ (C\ _) \rightarrow C$,
$(C\ _)\ (C\ _)\ (T\ _) \rightarrow C$,	$(C\ _)\ (C\ _)\ (C\ _) \rightarrow C$,	$(T\ _)\ (T\ _) \rightarrow T$,
$(T\ _)\ (C\ _) \rightarrow C$,	$(C\ _)\ (T\ _) \rightarrow C$,	$(C\ _)\ (C\ _) \rightarrow C$,
$(T\ _) \rightarrow T$,	$(C\ _) \rightarrow T$,	$\varnothing \rightarrow C$.

Nous arrêterons donc les stratégies comme liste de règles sans préciser leur nombre.

Partie entre deux stratégies

Ce sera une liste de couples (jeu1, jeu2) cette partie peut être limitée à un nombre fixe tel que 10 dans nos expérimentations ou 1000 pour [Delahaye, Mathieu 92] cependant que 100 ou 10 leur donnent des résultats très similaires. Par contre le nombre de coups -pivot semble être 4, signifiant que des parties à moins de 4 coups favorisent les stratégies "méchantes". Dans le but d'établir des comparaisons entre stratégies, les renoncements coûteront R=3 à chacun des adversaires, mais seront donc suivies d'un autre coup. Cette option est prise également pour ne pas avoir à considérer des "demi-renoncements" comme cela se produira en étendant la notion de trahison-coopération à une échelle continue sur [-1, 1].

II Mode d'inférence des stratégies vis à vis des jeux antérieurs

Métastratégie 1

Les individus sur lesquels nous travaillons sont des stratégies, nous nommons alors "métastratégie" l'algorithme permettant à ces "stratégies" de s'exercer.

La première de ces métastratégies est celle utilisée jusqu'à présent, conditionnée par le fait que les règles ne sont pas contradictoires, donc ne se chevauchent pas et décrivent complètement les situations possibles. A chaque étape une et une seule règle devant s'appliquer, détermine sans ambiguité la décision à prendre. Cependant, il convient d'adopter pour la suite, lors de génération automatique de stratégies une position à prendre : le conflit sera résolu en adoptant la première règle déclenchable, et d'autre part la décision sera le renoncement si aucune règle ne s'applique.

Métastratégie 2

L'algorithme de déroulement, et donc de construction d'une partie, décrit précédemment, s'apparente à celui d'un système expert déductif où la décision est donnée par une seule règle devant s'appliquer d'après les faits observés. L'algorithme du "contrôle flou" [Mamdani 74] permet également une décision en fonction des faits observés, mais en acceptant que plusieurs règles se chevauchent, que des situations ne soient pas représentées par une règle et que des règles de prémisses voisines puissent avoir des conclusions plus ou moins contradictoires. Le principe de cet algorithme est de confronter les conclusions et d'arriver à une décision en prenant le centre de gravité de leur union lorsqu'il s'agit d'ensembles flous, ou bien par un simple calcul de moyenne [Sugeno 85]. La décision finale est donc $(\sum \alpha_i d_i) / (\sum \alpha_i)$ où d_i est la conclusion d'une règle et α_i le degré de satisfaction de cette règle par les coups observés.

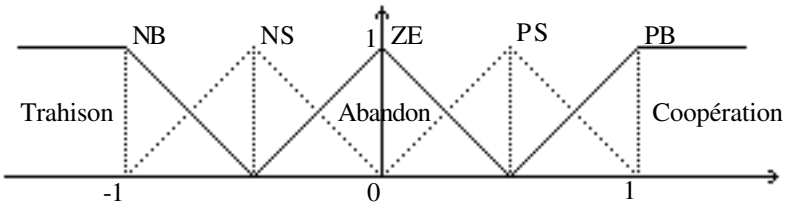
Si, donc, au lieu de valeurs symboliques T et C, on évalue par -1 la trahison, par 0 le renoncement et par 1 la coopération, un "coup" pouvant être un couple (x, y) de $[-1, 1]^2$, une partie étant toujours une liste de tels couples.

Une règle étant toujours de la forme :

(A1 M1) (A2 M2) (A3 M3) ... (An Mn) → Décision ∈ [-1, 1]

Deux options se présentent, la première consistant à maintenir les symboles T, R, C, également nommés NB, ZE, PB, et à rajouter les symboles NS (demi-trahison) et PS (demi-coopération) suivant la partition binaire habituelle de [-1, 1] en 5 prédicats flous. Ou bien, ce qui sera envisagé plus tard, de prendre pour prémisses des prédicats sous forme de nombres flous triangulaires (mode, rayon) que l'on fera varier et modifier grâce à des transitions génétiques de type bruitage.

La partition que nous utiliserons dans l'expérimentation est donnée par le schéma :



Degré de satisfaction d'une règle

Chaque prémisse (A M) sera confrontée au fait (Ja Jm), pour les trois valeurs exactes dont on a parlé initialement T = -1, R = 0, C = 1, les boules floues de rayon r (ici r = 1/2) autour de ces trois nombres constituent trois prédicats flous, et un fait exact ou non, confronté à un de ces prédicat donne un niveau de satisfaction définit par la fonction d'appartenance à ce prédicat

$$\mu_{réf}(\text{fait}) = \max [0, 1 - |\text{référence} - \text{fait}| / r]$$

Avec la convention 0/0 = 1, il est clair que l'on a une généralisation du cas exact si r ≤ 1. Par contre si r > 1 on pourrait avoir la référence 1, le fait 0 et une satisfaction 1/2 par exemple si r = 2 ce qui ferait une conclusion à moitié vérifiée même avec des faits exacts.

Cependant, dans la mesure où les règles ayant aucune, une ou deux prémisses devant s'appliquer dans le cas exact pour les premier, deuxième ou troisième coups, s'appliquent en se superposant aux autres dans les métastratégies 2 ou 3, alors il n'y a plus généralisation.

Le second problème qui se pose est qu'une règle va comporter généralement plus d'une prémisse, la conjonction est alors interprétée par une t-norme. Par exemple si seul le dernier coup (A M) figure dans la règle il faut évaluer "Ja est A" et "Jm est M" ce qui peut se faire par :

$\alpha = \max (0, 1 - |A - Ja| - |M - Jm|)$ en prenant r = 1/2 et la t-norme de Lukasiewicz ou :

$\alpha = \max (0, 1 - (|A - Ja| + |M - Jm|) / 4) \in [0, 1]$ si r = 2

$\alpha = \max (0, \min (1 - 2|A - Ja|, 1 - 2|M - Jm|))$ en prenant r = 1/2 et la t-norme min" de Zadeh.

Nous arrêtons le choix de prendre la famille habituelle de 5 prédicats (Y, R, C avec deux symboles supplémentaires) pour observer l'évolution des stratégies lors de l'apprentissage

génétique, et de prendre r = 1/2. La conjonction des prémisses est assurée par la t-norme "min", la conclusion est la moyenne des conclusions de toutes les règles qui se sont appliquées conformément à Sugeno. Notons à ce propos que nous réalisons un contrôleur flou dont les règles sont de longueur variables.

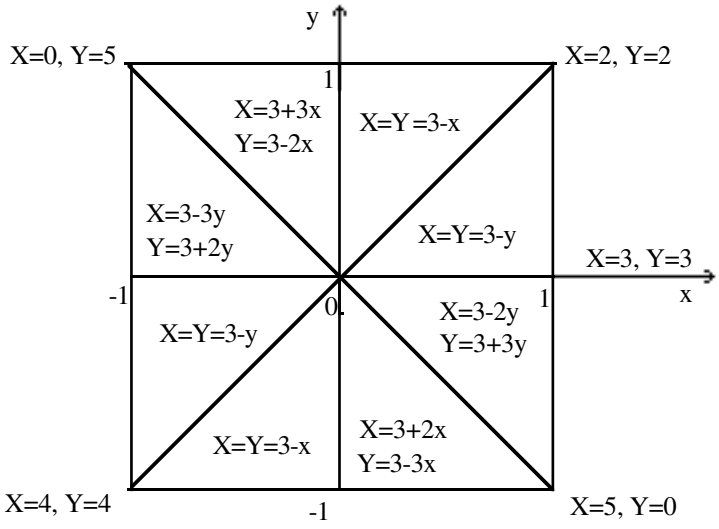
Métastratégie 3

C'est celle de Mamdani où le degré de satisfaction est calculé de la même façon par la fonction d'appartenance (min des deux fonctions) du couple (Ja, Jm) au précicat (A, M) de $[-1, 1]^2$, mais où la conclusion (dans la famille NB, NS, ZE, PS, PB) est tronquée conformément au modus ponens généralisé appliqué avec l'opérateur "min" de Mamdani pour l'implication. L'agrégation des conclusions est réalisée par la t-conorme "max", et la défuzzyfication est opéré comme l'abscisse du centre de gravité de l'union des conclusions. Insistons sur la différence essentielle avec les conclusions numériques de la méthode de Sugeno, une même règle s'appliquant plusieurs fois avec cette dernière, sera affectée d'un poids plus fort, alors qu'avec la méthode de Mamdani elle ne comptera toujours qu'une fois. C'est pourquoi, lors d'expérimentation génétique de contrôleur flou de Sugeno, il n'est pas rare de voir engendrer plusieurs fois la même règle traduisant une apparition naturelle de "coefficients" pour un corpus de règles.

Evaluation d'une stratégie

L'évaluation d'une partie entre S1 et S2 est la somme des coûts pour S1 comme pour S2 suivant la barème T = 0 < C = 2 < R = 3 < P = 4 < D = 5.

En ce qui concerne les deux autres métastratégies, il est nécessaire de définir une fonction de coût $(x, y) \in [-1, 1]^2 \rightarrow (X, Y)$ donnant les résultats d'un "coup" suivant ce qui a été joué x et y, et généralisant les définitions initiales (les quatre sommets du carré ci-dessous) ainsi que la convention supplémentaire X = Y = R sur chacun des axes. L'extention affine (par facettes) vérifiant ces contraites est donnée ici :



III L'apprentissage génétique

L'intérêt du flou, au cours de l'apprentissage, va être de mixer des stratégies régulières pour la première métastratégie, donc de constituer de nouvelles stratégies éventuellement incomplètes mais redondantes mettant en avant des règles par rapport aux autres, puis de former des stratégies possédant des labels tels que demi-trahison ou demi-coopération conservant leur sens intuitifs.

Dans [Axelrod 87, 92] une stratégie est représentée par une table de 64 symboles résultant des conclusions à jouer suivant les 4*4*4 possibilités des 3 coups précédents, plus les 6 = 3!

possibilités pour les 3 premiers coups, soit 70 gènes par chromosome. La stratégie optimale obtenue par algoritme génétique est en gros "donnant-donnant" [Michalewicz 92].

Dans [Delahaye, Mathieu 92] en mesurant les scores cumulés la meilleure stratégie est "donnant-donnant" suivie de "majorité-mou". Cette évolution se fait dans une population comportant initialement 100 individus du même type, soit 1200 individus. Le seul opérateur utilisé étant la reproduction, chacune des 12 stratégies initiales se reproduit au prorata de sa valeur cumulée face à la population courante. On observe alors une stabilisation vers la trentième génération.

Problèmes soulevés par l'évaluation des performances

Si nous cherchons l'évaluation d'une stratégie S comme la somme de ses coûts face à toutes les statégies de la population courante, cette définition est tout à fait originale par rapport à ce qui est fait habituellement dans les problèmes d'optimisation résolus par algorithmes génétiques, car de générations en générations cette fonction est modifiée et est donc à chaque étape une fonction de la population.

Cette définition présente un très gros inconvénient, à savoir que l'évaluation ne dépendant que de la population actuelle, il est impossible de faire des comparaisons. D'autre part, et c'est d'autant plus le cas si la population est réduite ou assez homogène, l'évaluation d'un individu n'est en fait que son évaluation relativement à son "voisinage".

C'est pourquoi nous avons d'abord expérimenté cette évaluation, y compris en renouvelant aléatoirement jusqu'au tiers de la population afin de lui donner un caractère plus universel, puis nous avons confronté les meilleurs individus à la population initiale, sans qu'il montrent alors une bonne performance.

Après une telle expérience infructueuse où les individus étaient évalués par rapport à la population courante, nous avons redéfini la valeur des stratégies par leur coût moyen face à la population (dite de référence) formée par 10 stratégies, calculées au moyen de parties de 10 coups face à chaque adversaire.

Le choix d'une population de référence suffisamment large et hétérogène reste bien sûr sujet à discussion, en fait un individu, quel que soit l'objectif fixé, doit être jugé sur ses aptitudes en toutes circonstances, c'est à dire ici, si le résultat moyen est bon lorsqu'il est confronté à un "grand nombre" d'autres stratégies très diversifiées. La solution de compromis que nous avons adopté est de prendre 10 stratégies intuitivement justifiables.

Caractéristiques de l'algorithme d'évolution employé

L'algorithme que nous utilisons ici, suit la méthode des algorithmes génétiques [Holland 75, Goldberg 90] mais avec certains apports déjà éprouvés avec succès [Gacôgne 93, 94]. Ces caractéristiques sont les suivantes :

a) Les chromosomes représentent un codage restant à un niveau concret (il s'agit donc plutôt de ce qu'on appelle les stratégies évolutives). Ils sont de longueurs variables et d'ailleurs structurés en règles comme listes de listes. L'idée étant déjà exprimée dans [Goldberg 89] mais surtout dans l'école allemande [Schwefel 90] et pour les arbres dans [Koza 92].

b) On évite les répétitions au sein de la population afin de maintenir la diversité, en cela on s'écarte des algorithmes décrits par [Goldberg 89].

c) Tous se reproduisent à chaque étape, ce qui permet de simuler une évolution plus rapide.

d) Lors de l'application d'un opérateur génétique, parents et enfants sont immédiatement triés et la famille est tronquée, cette solution permettant de ne garder que le dernier descendant d'une lignée amorçant une descente dans un puits de la fonction.

e) Le cross-over est d'un emploi tardif afin d'éviter une convergence trop rapide vers le même puits de la fonction de coût. Cette idée rejoint celle évoquée dans [Davis 91] diminuant la probabilité de mutation au cours de l'évolution. L'ensemble de tous les minimums pouvant être obtenus dans des "niches" [Horn 93].

f) Les transitions utilisées, outre la mutation et le cross-over, sont la suppression aléatoire de règle la naissance d'une nouvelle règle aléatoire, la recopie à droite ou à gauche d'un gène ou son remplacement par son suivant ou précédent dans un ordre donné sur l'ensemble des gènes.

g) Les transitions sont notées pour leur performance à chaque génération, une nouvelle transition de l'espèce de la meilleure étant créée à chaque génération. Au cas où aucune amélioration n'est observée, ainsi qu'au démarrage du processus, la population des transitions est recréée et brouillée aléatoirement. On évite ainsi une trop rapide convergence des transitions vers les cross-over ou une autre transition.

Choix d'une population de référence

En ce qui concerne, le mode d'inférence exact, nous avons testé les scores d'une population formée des stratégies gentille G, méchante M, donnant-donnant DD, périodique-gentil PG, périodique-méchant PM, majorité-mou MM, donnant-donnant-méfiant DDM, majorité-dur MD, donnant-donnant dur DDD et contrariant-gentil CG, cette dernière stratégie consistant à jouer la coopération au début, puis toujours le contraire du jeu précédent de l'adversaire.

En se limitant à des parties de 10 coups, les scores cumulés sur la population permettent un classement qui n'est pas véritablement celui :

$$DD_{2.43} < MM_{2.46} < PG_{2.72} < DDD_{2.72} < G_{2.875} < DDM_{3.09} < MD_{3.161} < M_{3.165} < PM_{3.23}$$

rencontré chez [Delahaye 92] où nous avons figuré les coûts moyens, il est vrai que nous n'avons pas pris en compte les stratégies aléatoire, sondeur et rancunière et que CG figure à leur place. Il est néanmoins curieux de constater que les stratégies DD et DDM qui ne diffèrent qu'en début de partie, se retrouvent pratiquement aux extrémités. Par ailleurs nos résultats sont sensiblement différents dans la mesure où DD n'est pas la meilleure stratégie. Nous obtenons pour la stratégie exacte :

$$DDD_{2.46} < DD_{2.55} < MM_{2.61} < M_{2.64} < PG_{2.66} < CG_{2.70} < PM_{2.91} < MD_{2.92} < G_{2.93} < DDM_{2.96}$$

Naturellement certains résultats sont très voisins, ce qui suggère qu'une meilleure appréciation doit être réalisée sur des parties plus longues, mais surtout face à une population plus importante et diversifiée.

En ce qui concerne la stratégie Sugeno, nous obtenons des valeurs proches mais avec un ordre sensiblement différent :

$$DDD_{2.61} < DD_{2.66} < CG_{2.69} < M_{2.72} < MM_{2.73} < PM_{2.80} < G_{2.85} < DDM_{2.91} < MD_{2.97} < PG_{3.01}$$

Quant à l'algorithme de Mamdani, il donne exactement le même ordre :

$$DDD_{2.67} < DD_{2.71} < CG_{2.73} < M_{2.74} < MM_{2.78} < PM_{2.87} < G_{2.87} < DDM_{2.92} < MD_{2.96} < PG_{3.00}$$

IV Expérimentation

IV - 1 Cas exact, évolution observée à partie des décisions possibles C ou T

A partir d'une population initiale aléatoire, en 20 générations la valeur moyenne 2.13 est obtenue avec les règles plutôt méchantes :

$$(_\ _)\ (T\ T) \rightarrow T \qquad (C\ _) \rightarrow T \qquad \varnothing \rightarrow T$$

IV - 2 Cas exact, évolution observée à partie des décisions possibles C, R ou T

Pour une population initiale aléatoire, R n'intervient pas (il intervient cependant par défaut puisque les règles sont incomplètes) dans la solution trouvée en 13 générations de valeur 2.24 (valeur 2.22 en Sugeno) :

$$(_\ C) \rightarrow T \qquad (C\ _) \rightarrow T \qquad \varnothing \rightarrow T$$

IV - 3 Cas Sugeno, décisions possibles T=NB, NS (demi-trahison), R=ZE (passer son tour), PS (demi-coopération) ou C=PB (coopération)

Rappelons que la décision obtenue comme moyenne des conclusions de règles, est une valeur entre -1 et 1. En partant de la population de référence, en 12 générations, cette valeur a été abaissée à 2.22 pour la stratégie : (R _) → R (C _) → T ∅ → T

Puis à 2.21 grâce à (R _) → PS (C _) → T ∅ → T

A 2.13 pour : (R _) → C (C _) → T ∅ → T

A 2.06 en 55 générations avec :
(R T) → R (R C) → C (C _) → T ∅ → T

Enfin à 2.03 (valeur 2.27 en Mamdani) au bout de 135 générations avec le jeu CM1 de règles :
(T R) → C (R _) → C (NS T) → R
(R T) → R (PS T) → NS (C _) → T ∅ → T

Partant cette fois d'une population aléatoire, nous observons vers la 60° génération la solution de valeur 2.13 déjà rencontrée, puis à la génération 280, la valeur 1.98 (valeur 2.34 en Mamdani) pour les 5 règles ne faisant pas intervenir de demi-décisions (stratégie CM2) :
(_ _) (T T) → C (R C) → C
(R R) (R T) → T (C _) → T ∅ → T

On remarquera qu'hormis quelques règles la conclusion est l'opposée du jeu précédent de l'adversaire. La dernière règle s'applique entièrement à chaque coup et n'est vraiment contrebalancée que si l'adversaire hésite entre passer ou trahir, mais elle se trouve renforcée si l'adversaire a coopéré. A cause de ces remarques, nous nommons le prototype de ces stratégies contrariante-méchante (CM).

IV - 4 Cas Mamdani, décisions possibles T=NB, NS, R=ZE, PS ou C=PB

Le calcul de la valeur d'une stratégie suivant cet algorithme étant beaucoup plus long, nous n'avons réalisé que deux sessions, néanmoins fructueuses, puisque montrant des stratégies retrouvant le même genre de règles. A partir de populations aléatoires nous obtenons vers la 30° génération :
2.19 pour :
(T R) → C (R C) → C (R T) → R (C _) → T ∅ → T

Puis 2.16 (valeur 2.28 en Sugeno) pour le système que nous nommons CM3 :
(_ R) → C (R PS) → C (R NS) → R
(PS C) (PS R) → PS (C _) → T ∅ → T

On remarquera que les jeux de règles CM1 et CM2 inversent leurs performances suivant les deux métastratégies employées.

Variation du rayon des prédicats triangulaires

Si nous considérons la meilleur stratégie obtenue CM2 suivant Sugeno :

(_ _) (T T) → C (R C) → C
(R R) (R T) → T (C _) → T ∅ → T

Son score 1.98 est conservé pour $0 < r < 2/3$ (règles se chevauchant pas ou peu), puis il s'améliore brusquement à la valeur 1.95 entre 2/3 et 1(les règles se chevauchant davantage), puis passe par un maximum 2.41 pour $r = 1.3$ à 1.5 et un minimum 2.33 pour $r = 2$.

Conclusion

Les premières stratégies apparues qualifiées de contrariantes ou contrariantes-méchantes le sont à cause des choix initiaux sur les coûts de la coopération, trahison et abandon. Il faut mentionner à ce propos que l'abandon n'est en fait que "passer son tour". Il se peut que relever ce dernier coût change l'allure des jeux de règles obtenus vers une plus grande coopération. En fait les expériences réalisées dans ce sens [Delahaye 92] n'ont pas montré de différences significatives. Les résultats observés ici sont plus le fait que la stratégie d'application des stratégies n'est plus la même, puisqu'on prend la décision résultant de la moyenne des règles déclenchées suivant la méthode de Sugeno.

La principale remarque à faire sur les solutions qui émergent d'un processus d'évolution où la possibilité est donnée de prendre des décisions floues, est que cette évolution semble écarter le plus souvent les règles mettant en oeuvre des demi-mesures. Généralement les positions sont donc bien tranchées et l'observation pas à pas des parties itérées montre que la décision est presque toujours T, R ou C.

Par la suite, nous pourrons envisager des transitions génétiques comme des "bruits" modifiant les modes et la largeur r des prédicats, quoiqu'il semble préférable de maintenir une représentation symbolique des décisions afin de mieux interpréter les solutions engendrées.

Références

Axelrod R. The genetic algorithm for the prisoner dilemma problem. pp32-41 in Genetic algorithms and simulated annealing, Morgan Kaufmann Pub. 1987

Axelrod R. Donnant-donnant, théorie du comportement coopératif, Ed. Odile Jacob 1992

Davis T.E. Principe J.C. A simulated annealing like convergence theory for the simple genetic algorithm, Proceedings of the 4th International Conference on G.A. p.174-182 1991

Delahaye J.P. L'altruisme récompensé, Pour la science novembre 1992

Delahaye J.P. Mathieu P. Expériences sur le dilemme itéré des prisonniers. Rapport 233 du Laboratoire d'Informatique fondamentale de Lille 1992

Delahaye J.P. Mathieu P. L'altruisme perfectionné. Rapport 249 du LIFL 1993

Gacôgne L. Compte rendu sur la recherche de table de contrôleur flou par algorithme génétique, Rapport 93/26 du Laforia-Université paris VI 1993

Gacôgne L. Apprentissage génétique global d'un contrôleur flou à deux variables basé sur la simulation d'un véhicule autonome, Actes de la conférence IPMU 1994

Goldberg D.E. Genetic algorithms in search optimization and machine learning, Ad.Weisley 89

Goldberg D.E. Korb B. Deb K. Messy genetic algorithms : motivation, analysis and first results. Complex systems n°3 p.493-530 1989

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

Holland J. Une échappatoire à la précarité : les possibilités de l'application des algorithmes généraux d'apprentissage aux systèmes parallèles à base de règles, Apprentissage symbolique p.523-554 Cépaduès 1993

Horn J. Finite Markov chain analysis of genetic algorithms with niching, Report 93/002 University of Illinois at Urbana Champaign 1993

Mamdani E.H. Assilian S. Learning control algorithm in real dynamic systems. Proceedings of the 4th IFAC IFIP Zürich 1974

Michalewicz Z. Genetic algorithms+data structures = evolution programs, Springer Verlag 1992

Schwefel H.P. Systems analysis, systems design, and evolutionary strategies. Systems analysis, Modeling and Simulation vol 7 (11/12) p.853-864 1990

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

Listing d'une partie entre CM1 et MD

(partie CM1 MD 10 'Sugeno) ; Les règles ne s'appliquant pas ne sont pas éditées
((c (t) (ze)) (c (ze) (any)) (ze (ze) (t)) (ze (ns) (t)) (ns (ps) (t)) (t (c) (any)) (t)) joue contre ((t (t t t) (any any any)) (t (t t c) (any any any)) (t (t c t) (any any any)) (t (c t t) (any any any)) (c (t c c) (any any any)) (c (c t c) (any any any)) (c (c c t) (any any any)) (c (c c c) (any any any)) (t (t t) (any any)) (c (c c) (any any)) (t (c t) (any any)) (t (t c) (any any)) (c (c) (any)) (t (t) (any)) (t))
Règle : adv. () moi () --> t Coeff. 1
decision : -1 ; c'est la décision de CM1
Règle : adv. () moi () --> t Coeff. 1
decision : -1 ; c'est la décision de MD
Coup 1 jeux -1 et -1 Gains (4 4)
décision : 0
Règle : adv. (t) moi (any) --> t Coeff. 1
decision : -1
Coup 2 jeux 0 et -1 Gains (3 3)
Règle : adv. (t) moi (ze) --> c Coeff. 1
decision : 1
décision : 0 ; aucune règle ne s'est appliquée pour MD
Coup 3 jeux 1 et 0 Gains (3 3)
Règle : adv. (ze) moi (any) --> c Coeff. 1
decision : 1
Règle : adv. (c) moi (any) --> c Coeff. 1
decision : 1
Coup 4 jeux 1 et 1 Gains (2 2)
Règle : adv. (c) moi (any) --> t Coeff. 1
decision : -1
Règle : adv. (c c) moi (any any) --> c Coeff. 1
decision : 1
Coup 5 jeux -1 et 1 Gains (0 5)
Règle : adv. (c) moi (any) --> t Coeff. 1
decision : -1
Règle : adv. (t c c) moi (any any any) --> c Coeff. 1
Règle : adv. (t c) moi (any any) --> t Coeff. 1
Règle : adv. (t) moi (any) --> t Coeff. 1 ; ici la trahison compte 2 fois plus que la coopération
decision : -3.333333E-1
Coup 6 jeux -1 et -3.333333E-1 Gains (3.333333 3.333333)
Règle : adv. (ze) moi (any) --> c Coeff. 3.333333E-1
Règle : adv. (ze) moi (t) --> ze Coeff. 3.333333E-1
Règle : adv. (ns) moi (t) --> ze Coeff. 6.666667E-1 ; ici 3 règles se chevauchent
decision : 2.500000E-1
Règle : adv. (t t c) moi (any any any) --> t Coeff. 1
Règle : adv. (t t) moi (any any) --> t Coeff. 1
Règle : adv. (t) moi (any) --> t Coeff. 1 ; la trahison est 3 fois déduite
decision : -1
Coup 7 jeux 2.500000E-1 et -1 Gains (3.5 2.25)
Règle : adv. (t) moi (ze) --> c Coeff. 5.000000E-1
decision : 1
décision : 0
Coup 8 jeux 1 et 0 Gains (3 3)
Règle : adv. (ze) moi (any) --> c Coeff. 1
decision : 1
Règle : adv. (c) moi (any) --> c Coeff. 1
decision : 1
Coup 9 jeux 1 et 1 Gains (2 2)
Règle : adv. (c) moi (any) --> t Coeff. 1
decision : -1
Règle : adv. (c c) moi (any any) --> c Coeff. 1
Règle : adv. (c) moi (any) --> c Coeff. 1
decision : 1
Coup 10 jeux -1 et 1 Gains (0 5)
Résultats (2.383333E1 3.258334E1)

Listing de l'évaluation de CM1 suivant Sugeno

((c (t) (ze)) (c (ze) (any)) (ze (ze) (t)) (ze (ns) (t)) (ns (ps) (t)) (t (c) (any)) (t)) joue contre ((t (t) (any)) (c (c) (any)) (c)) DD
Coup 1 jeux -1 et 1 Gains (0 5)
Coup 2 jeux -1 et -1 Gains (4 4)
Coup 3 jeux 0 et -1 Gains (3 3)
Coup 4 jeux 1 et 0 Gains (3 3)
Coup 5 jeux 1 et 1 Gains (2 2)
Coup 6 jeux -1 et 1 Gains (0 5)
Coup 7 jeux -1 et -1 Gains (4 4)
Coup 8 jeux 0 et -1 Gains (3 3)
Coup 9 jeux 1 et 0 Gains (3 3)
Coup 10 jeux 1 et 1 Gains (2 2)
Résultats (24 34)
CM1 joue contre ((t (t t) (any any)) (t (t c) (any any)) (t (c t) (any any)) (c (c c) (any any)) (c (any) (any)) (c)) DDD
Coup 1 jeux -1 et 1 Gains (0 5)
Coup 2 jeux -1 et 1 Gains (0 5)
Coup 3 jeux -1 et 0 Gains (3 3)
Coup 4 jeux 0.5 et 0 Gains (3 3)
Coup 5 jeux 1 et 1 Gains (2 2)
Coup 6 jeux -1 et 1 Gains (0 5)
Coup 7 jeux -1 et 0 Gains (3 3)
Coup 8 jeux 0.5 et 0 Gains (3 3)
Coup 9 jeux 1 et 1 Gains (2 2)
Coup 10 jeux -1 et 1 Gains (0 5)
Résultats (16 36)
CM1 joue contre ((c (c) (any)) (t (t) (any)) (t)) DDM
Coup 1 jeux -1.000000E0 et -1 Gains (4 4)
Coup 2 jeux 0 et -1 Gains (3 3)
Coup 3 jeux 1 et 0 Gains (3 3)
Coup 4 jeux 1 et 1 Gains (2 2)
Coup 5 jeux -1 et 1 Gains (0 5)
Coup 6 jeux -1 et -1 Gains (4 4)
Coup 7 jeux 0 et -1 Gains (3 3)
Coup 8 jeux 1 et 0 Gains (3 3)
Coup 9 jeux 1 et 1 Gains (2 2)
Coup 10 jeux -1 et 1 Gains (0 5)
Résultats (24 34)
CM1 joue contre ((t (any) (any)) (t)) Méchant
Coup 1 jeux -1 et -1 Gains (4 4)
Coup 2 jeux 0 et -1 Gains (3 3)
Coup 3 jeux 1 et -1 Gains (5 0)
Coup 4 jeux 0 et -1 Gains (3 3)
Coup 5 jeux 1 et -1 Gains (5 0)
Coup 6 jeux 0 et -1 Gains (3 3)
Coup 7 jeux 1 et -1 Gains (5 0)
Coup 8 jeux 0 et -1 Gains (3 3)
Coup 9 jeux 1 et -1 Gains (5 0)
Coup 10 jeux 0 et -1 Gains (3 3)
Résultats (39 19)
CM1 joue contre ((c (any) (any)) (c)) Gentil
Coup 1 jeux -1 et 1 Gains (0 5)
Coup 2 jeux -1 et 1 Gains (0 5)
Coup 3 jeux -1 et 1 Gains (0 5)
Coup 4 jeux -1 et 1 Gains (0 5)
Coup 5 jeux -1 et 1 Gains (0 5)
Coup 6 jeux -1 et 1 Gains (0 5)
Coup 7 jeux -1 et 1 Gains (0 5)
Coup 8 jeux -1 et 1 Gains (0 5)
Coup 9 jeux -1 et 1 Gains (0 5)
Coup 10 jeux -1 et 1 Gains (0 5)
Résultats (0 50)
CM1 joue contre ((c (t) (any)) (t (c) (any)) (c)) CG
Coup 1 jeux -1 et 1 Gains (0 5)
Coup 2 jeux -1 et 1 Gains (0 5)
Coup 3 jeux -1 et 1 Gains (0 5)
Coup 4 jeux -1 et 1 Gains (0 5)
Coup 5 jeux -1 et 1 Gains (0 5)
Coup 6 jeux -1 et 1 Gains (0 5)
Coup 7 jeux -1 et 1 Gains (0 5)
Coup 8 jeux -1 et 1 Gains (0 5)
Coup 9 jeux -1 et 1 Gains (0 5)

Coup 10 jeux -1 et 1 Gains (0 5)
Résultats (0 50)
CM1 joue contre ((c (any any) (t c)) (c (any any) (c t)) (t (any any) (c c)) (c (any) (c)) (c)) PG
Coup 1 jeux -1 et 1 Gains (0 5)
Coup 2 jeux -1 et 1 Gains (0 5)
Coup 3 jeux -1 et 0 Gains (3 3)
Coup 4 jeux 0.5 et 0 Gains (3 3)
Coup 5 jeux 1 et 0 Gains (3 3)
Coup 6 jeux 1 et 0 Gains (3 3)
Coup 7 jeux 1 et 0 Gains (3 3)
Coup 8 jeux 1 et 0 Gains (3 3)
Coup 9 jeux 1 et 0 Gains (3 3)
Coup 10 jeux 1 et 0 Gains (3 3)
Résultats (24 34)
CM1 joue contre ((t (any any) (t c)) (t (any any) (c t)) (c (any any) (t t)) (t (any) (t)) (t)) PM
Coup 1 jeux -1 et -1 Gains (4 4)
Coup 2 jeux 0 et -1 Gains (3 3)
Coup 3 jeux 1 et 0 Gains (3 3)
Coup 4 jeux 1 et 0 Gains (3 3)
Coup 5 jeux 1 et 0 Gains (3 3)
Coup 6 jeux 1 et 0 Gains (3 3)
Coup 7 jeux 1 et 0 Gains (3 3)
Coup 8 jeux 1 et 0 Gains (3 3)
Coup 9 jeux 1 et 0 Gains (3 3)
Coup 10 jeux 1 et 0 Gains (3 3)
Résultats (31 31)
CM1 joue contre ((t (t t t) (any any any)) (t (t t c) (any any any)) (t (t c t) (any any any)) (t (c t t) (any any any)) (c (t c c) (any any any)) (c (c t c) (any any any)) (c (c c t) (any any any)) (c (c c c) (any any any)) (t (t t) (any any)) (c (c c) (any any)) (c (c t) (any any)) (c (t c) (any any)) (c (c) (any)) (t (t) (any)) (c)) MM
Coup 1 jeux -1 et 1 Gains (0 5)
Coup 2 jeux -1 et -1 Gains (4 4)
Coup 3 jeux 0 et -1 Gains (3 3)
Coup 4 jeux 1 et 0 Gains (3 3)
Coup 5 jeux 1 et 1 Gains (2 2)
Coup 6 jeux -1 et 1 Gains (0 5)
Coup 7 jeux -1 et 3.333333E-1 Gains (2 3.666667)
Coup 8 jeux -2.235174E-8 et -1 Gains (3 3)
Coup 9 jeux 1 et 0 Gains (3 3)
Coup 10 jeux 1 et 1 Gains (2 2)
Résultats (22 33.66666)
CM1 joue contre ((t (t t t) (any any any)) (t (t t c) (any any any)) (t (t c t) (any any any)) (t (c t t) (any any any)) (c (t c c) (any any any)) (c (c t c) (any any any)) (c (c c t) (any any any)) (c (c c c) (any any any)) (t (t t) (any any)) (c (c c) (any any)) (t (c t) (any any)) (t (t c) (any any)) (c (c) (any)) (t (t) (any)) (t)) MD
Coup 1 jeux -1 et -1 Gains (4 4)
Coup 2 jeux 0 et -1 Gains (3 3)
Coup 3 jeux 1 et 0 Gains (3 3)
Coup 4 jeux 1 et 1 Gains (2 2)
Coup 5 jeux -1 et 1 Gains (0 5)
Coup 6 jeux -1 et -3.333333E-1 Gains (3.333333 3.333333)
Coup 7 jeux 2.500000E-1 et -1 Gains (3.5 2.25)
Coup 8 jeux 1 et 0 Gains (3 3)
Coup 9 jeux 1 et 1 Gains (2 2)
Coup 10 jeux -1 et 1 Gains (0 5)
Résultats (23.83333 32.58334)
Moyenne = 2.03

Listing des modules

```
; GENERATIONS
(de generation (P OP) ; renvoie la liste dont le car est le nouveau P et le cdr, le nouvel OP triés.
      (gen1 nil P nil OP 0))
(de gen1 (PV PR OPV OPR k) (cond ; parties vues et restantes de P et OP, k est un indice
      ((or (null OPR) (null PR) (eq k np)) (cons (firstn np (tri (elim (append PV PR))))
                                                 (genop (firstn nop (tri (append OPV OPR))))))
       ((member (cadar OPR) '(crossover crossover1 crossover2))
          (gen4 (car OPR) (car PR)  (if (and PR (< k (div np 2))) (hasard PR) (hasard PV))  PV (cdr PR) OPV (cdr OPR) (1+ k)))
      (t ; (print (cadar opr) '" agit sur " (cdar PR))
      (gen2 (car OPR) (regul (funcall (cddar OPR) (cdar PR))) (cons (car PR) PV) (cdr PR) OPV (cdr OPR) (1+ k))) ))
(de gen2 (op res PV PR OPV OPR k) ; effectue les évaluations sur les résultats "res" obtenus par "op"
      ;(print '"résultat " res)
      (if (or (null res)(equal (cdar PV) res)) (gen1 PV PR (cons op OPV) OPR (1+ k)); on continue si res = op(res)
            (gen3 op (cons (valeur res) res) PV PR OPV OPR k))); pas le crossover
(de gen3 (op res PV PR OPV OPR k) ; évalue l'opérateur "op" suivant les "valeurs" en tête de "res"
      (print (cadr op) (car PV)'"-->" res); autres cas que le cross-over
      ; ici on ne garde que le meilleur entre le père et le fils
      (gen1 (if (< (caar PV) (car res)) PV (cons res (cdr PV)))
         PR  (cons (cons (+ (car op) (sgn (- (car res) (caar PV)))) (cdr op)) OPV) OPR k))
(de gen4 (op C1 C2 PV PR OPV OPR k)
      (gen5 op (funcall (cddr op) (cdr C1) (cdr C2)) C1 C2 (ret C2 PV) (ret C2 PR) OPV OPR k))
(de gen5 (op res C1 C2 PV PR OPV OPR k); évalue les résultats du cross-over
     (if (or (null C2)(null (car res)) (null (cdr res))) (gen1 PV PR (cons op OPV) OPR k)
        (gen6 op (cons (valeur (car res)) (car res)) (cons (valeur (cdr res)) (cdr res))  C1 C2 PV PR OPV OPR k)))
(de gen6 (op C1r C2r C1 C2 PV PR OPV OPR k) ; évalue "op" qui est un cross-over
      (print (cadr op) C1 C2 '"-->" C1r C2r)
       (gen1 (append (firstn 2 (tri (list C1 C2 C1r C2r))) PV) PR ; les 2 meilleurs chromosomes dans PV
           (cons (cons (+ (car op) (sgn (- (+ (car C1r) (car C2r)) (+ (car C1) (car C2)))))  (cdr OP)) OPV) OPR k))
(de genop (L)  ; renvoie la liste d'opérateurs L précédée d'un nouvel opérateur du type du premier de L
      (if (<= 0 (caar L)) L
           (print '"Une nouvelle " (cadar L) '" est créée.")
           (cons (mcons 0 (cadar L) (funcall (cadar L))) L) ))
(de sgn (x) (cond ((< 0 x) 1) ((> 0 x) -1) (t 0))) ; trois sortes de signes
(de ret (X L) (cond ((null L) L) ((equal X (car L)) (cdr L)) (t (cons (car L) (ret X (cdr L)))))));retire la 1° occurrence de X dans L
(de elim (L) (cond ; permet d'éliminer les répétitions de L c'est à dire de ne conserver que des individus distincts
      ((null L) L)   ; élimination faible
      ((null (car L)) (elim (cdr L)))
      ((member (car L) (cdr L)) (elim (cdr L)))
      (t (cons (car L) (elim (cdr L)))) ))
(de enchaine (n) ; enchaîne n générations en modifiant les globaux P, OP et STAT, lit np, nt et nop
    (let ((num 0) (etat nil) )(until (eq num n); (eq (caar P) 0)) teste si on est est sûr du coût 0 optimal
        (set 'etat (generation P OP))  (set 'P (car etat)) ; état est constitué par ((P) OP)
        (set 'stat (cons (list (caar P)(car (nth (- np nt) P))) stat)) ; STAT conserve les performances extrèmes
        (set 'P (append (firstn (- np nt) P) (valoriser
                    (produc (+ nt (- np (length P))))))) ; permet de rajouter NT individus aléatoires
       (set 'OP (if (<= 0 (caaddr etat))  (initrans nop) (cdr etat)))
        (set 'num (1+ num)) (print '"La génération " num '" est achevée.  Valeurs : "
           (mapcar 'car P) '" Transitions : " (mapcar 'cadr OP))) ) ))
; TRI
(de fusion (L1R L2R LF) (cond ; où chaque élément de LR est de la forme (clé ... code de l'état...)
      ((null L1R) (if (null L2R) (reverse LF) (fusion nil (cdr L2R) (cons (car L2R) LF))))
      ((null L2R) (fusion nil (cdr L1R) (cons (car L1R) LF)))
      ((< (caar L1R) (caar L2R)) (fusion (cdr L1R) L2R (cons (car L1R) LF)))
      (t (fusion L1R (cdr L2R) (cons (car L2R) LF))) ))
(de sep (pair LP LI LR) ;liste des éléments de rangs pairs, impairs et restant à séparer
      (if LR (if pair (sep nil (cons (car LR) LP) LI (cdr LR))
                      (sep t LP (cons (car LR) LI) (cdr LR)))
             (cons LP LI)) ) ; renvoie ((a0 a2 a4 ...) a1 a3 a5 ....)
(de tri (L) (if (null (cdr L)) L (tribis (sep t nil nil L))))
(de tribis (LC) (fusion (tri (car LC)) (tri (cdr LC)) nil))
; DECISION FLOUE (SUGENO ou MAMDANI) ou EXACTE, SYMBOLIQUE ou NUMERIQUE
(de sommet (S) ; donne l'abscisse du sommet du prédicat
   (if (numberp S) S (selectq  S (NB -1) (NS -0.5) (ZE 0) (R 0) (PS 0.5) (PB 1) (C 1) (T -1))))
(de opp (pr) ; donne l'opposé du prédicat pr
      (selectq  pr (NB 'PB) (NS 'PS) (ZE 'ZE) (ANY 'ANY) (PS 'NS) (PB 'NB)))
(de compte (n v) (if (eq n 0) nil (cons v (compte (1- n) v))))
(de histog (E) (cond; E est le nom d'un prédicat que l'on modifie en histogramme de 21 valeurs entre -1 et 1
      ((eq E 'ANY) (compte 21 1))
      ((eq E 'NUL) (compte 21 0))
       (t (let ((S (sommet E)) (i 1) (H '(1))) (repeat 20 (setq i (- i 0.1) H (cons i H)))
```

```
        (mapcar (lambda (x) (maxi 0 (- 1 (divide (abs (- S x)) r)))) H)))))
(de maxi (L M)  (cond ; réalise la liste composée des max dans les listes L et M ou bien le max de 2 nombres
      ((null L) M)
      ((null M) L)
      ((numberp L) (if (< L M) M L))
      ((< (car L) (car M)) (cons (car M) (maxi (cdr L) (cdr M))))
      (t (cons (car L) (maxi (cdr L) (cdr M))))))
(de tronc (h  ens) (cond ; renvoie l'ensemble flou tronqué à la hauteur h
      ((null ens) nil)
       ((atom ens) (tronc h (histog ens)))
      (t (cons (if (< (car ens) h) (car ens) h) (tronc h (cdr ens))))))
(de gravite (L) (gbis L -1 0 0)) ; donne l'abscisse du barycentre de la distribution L
; attention le centre de gravité de PB est 0.866 ou alors il faut définir PB comme un triangle.
(de gbis (LR k S pr ) (if (null LR) (if (eq pr 0) 0 (divide S pr))
                          (gbis (cdr LR) (+ k 0.1)(+ (* k (car LR)) S) (+ pr (car LR)))))
(de mini (a b) (if (< a b) a b))
;Une règle est (C Pa Pm) où C est la conclusion, Pa la liste des jeux précedents de l'adversaire, et Pm
; la liste de mes jeux précédents
(de coef (Ea Em Pa Pm) ; coef d'application de la règle de liste de prémisses Pa Pm pour l'entrée E = (e1 e2 ....)
      (if (or (null Pa) (null Pm)) (if (or (null Ea) (null Em)) 1 0); début de parties
          (coefbis Ea Em Pa Pm 1)))
(de coefbis (Ea Em Pa Pm co) (cond ; LC liste de coefficients
      ((null Ea) (if (null Pa) co 0))
      ((null Pa) co)
      ((null Em) (if (null Pm) co 0))
      ((null Pm) co)
      (t (coefbis (cdr Ea) (cdr Em) (cdr Pa) (cdr Pm) (mini co  (mini (poss (car Ea) (car Pa)) (poss (car Em) (car Pm)))))))
(de poss (v Pr) (cond; donne la satisfaction de la valeur (ou symbole) v pour le prédicat Pr
      ((eq Pr 'ANY) 1)
      ((eq Pr 'NUL) 0)
      (t (maxi 0 (- 1 (divide (abs (- (sommet Pr) (sommet v))) r))))))

(de decision (Ea Em LR mode) ; donne une sortie dans [-1, 1] avec S liste de règles
(if (null LR) 'R ; cas exceptionnel ou une base de règles exactes est incomplète, on passe son tour
    ;(prin '"Règle : adv. " (cadar LR) '" moi " (caddar LR) '" --> " (caar LR) '" Coeff. ")
          (decisionbis (coef Ea Em (cadar LR) (caddar LR)) Ea Em (caar LR)
                       (if (eq mode 'sugeno) 0 (histog 'nul)) 0 (cdr LR) mode)))
(de decisionbis (co Ea Em conc S SC LR mode) ;(print co); c coeff E les entrées, S sortie, SC somme des coef
      (if (eq mode 'exact) (if (equal co 1) conc (decision Ea Em LR mode))
      (decisionter Ea Em (if (eq mode 'sugeno) (+ (* co (sommet conc)) S) (maxi (tronc co conc) S)) (+ co SC) LR mode)))
(de decisionter (Ea Em S SC LR mode) ; S sortie
      (if (null LR) (cond       ((eq SC 0) 0)
                                ((eq mode 'sugeno) (divide S SC))
                                (t (gravite S)))
         ;(prin '"Règle : adv. " (cadar LR) '" moi " (caddar LR) '" --> " (caar LR) '" Coeff. ")
        (decisionbis (coef Ea Em (cadar LR) (caddar LR)) Ea Em (caar LR) S SC (cdr LR) mode)))
(de cout (x y) (cond
      ((not (numberp x)) (cout (sommet x) y))
      ((not (numberp y)) (cout x (sommet y)))
      (t (coutbis (cond         ((< 0 (* x y)) (if (< 0 x) (if (< x y) (- 3 x) (- 3 y))   (if (< x y) (- 3 y) (- 3 x))))
                                ((< 0 (+ x y)) (if (< 0 x) (list (* -2 y) (* 3 y)) (list (* 3 x) (* -2 x))))
                                ((<= 0 x) (list (* 2 x) (* -3 x)))
                                (t (list (* -3 y) (* 2 y))))))))
(de coutbis (L) (if (atom L) (list L L) (list (+ 3 (car L)) (+ 3 (cadr L)))))
```

; PARTIES de nc coups entre deux stratégies

```
(de partie (S1 S2 n mode) (print S1 '" joue contre " S2); renvoie le couple (C1 C2) des coûts pour les deux stratégies
      (par1 S1 S2 n mode nil nil 0 0))
(de par1 (S1 S2 n mode E1 E2 C1 C2) (if (zerop n) (print '"Résultats "(list C1 C2))
     (par2 S1 S2 (1- n) mode E1 E2 C1 C2 (decision E2 E1 S1 mode) (decision E1 E2 S2 mode))))
(de par2 (S1 S2 n mode E1 E2 C1 C2 J1 J2)  (prin '"Coup " (- nc n) '" jeux " J1 '" et " J2)
      (par3 S1 S2 n mode (cons J1 E1) (cons J2 E2) C1 C2 (cout J1 J2)))
(de par3 (S1 S2 n mode E1 E2 C1 C2 C)  (print '" Gains " C)
     (par1 S1 S2 n mode E1 E2 (+ C1 (car C)) (+ C2 (cadr C))))
```

; VALEUR D'UNE STRATEGIE (P est une liste globale de stratégies précédées de leurs valeurs)

```
(de valeur (S) (valbis S P0 0 0)) ; on fait appel à la population P ou P0 globale, donc l'ancienne
(de valbis (S LP val nb) (if   (null LP) (+ 0 ; (length S) ; (nbtotal S) ; on peut favoriser les énoncés les plus courts
                                    (truncate (* 100 (divide val (* nc nb)))))
                               (valbis S (cdr LP) (+ val (car (partie S (cdar LP) nc mode))) (1+ nb))))
(de valoriser (LI) (tri (mapcar (lambda (x) (cons (valeur x) x)) LI)))
(de nbtotal (L) (cond          ((null L) 0)  ; donne le cardinal de l'arbre
                               ((atom L) 1)
```

```
                        (t (+ (nbtotal (car L)) (nbtotal (cdr L))))))
; INITIALISATION de P et OP
;Un chromosome est une liste de la forme (R1 R2 R3 .... ) où R = (conc (c1 c2 ... cn...) (p1 p2 ..... pn...))
(de hasard (L) ; renvoie un élément tiré au hasard dans la liste L
      (nth (random 0 (length L)) L))
(de poisson (m) ; renvoie un entier au hasard suivant une loi de Poisson de moyenne m
       (poissonbis (exp (- m)) 0 (divide (random 0 100) 100)))
(de poissonbis (ex n x) ; vérifie s'il faut continuer à tirer des nombres au hasard
      (if (< x ex) n ; cas où le produit des x est parvenu à dépasser ex = exp(-m)
       (poissonbis ex (1+ n) (* x (divide (random 0 100) 100))) ))
(de nouvregle (m) (let ((k (poisson m)) (r1 nil) (r2 nil))  ; produit une nouvelle règle
      (repeat k (set 'r1 (cons (hasard PRED) r1)))
      (repeat k (set 'r2 (cons (hasard PRED) r2)))  (mcons (hasard (cdr PRED)) r1 (list r2))))
(de regulbis (R) (cond
     ((atom R) R)
     ((eq 'ANY (car R)) (cons (hasard (cdr PRED)) (cdr R)))
     (t R)))
(de regul (C) (mapcar 'regulbis C))
(de produc (k) (let ((pop nil)(c nil)) ; produit une population de k jeux de règles, c est un chromosome
       (repeat k (set 'c nil) (repeat (poisson 5) (set 'c (cons (nouvregle M) c)))
       (set 'pop (cons c pop)) (print '"Nouvelle solution aléatoire " c)) pop))
(de initrans (n) (print '"On réinitialise les opérateurs.") (set 'OP nil) ; donne n transitions
      (repeat (div n 7)(set 'OP (append OP (mapcar (lambda (f) (mcons 0 f (funcall f)))
            '(mutation crossover1 suppression suivant transpo precedent creation )))))
      (set 'OP (mapcar 'cdr (tri (mapcar (lambda (x)(cons (random 0 99) x)) OP)))))

(setq PRED '(ANY t NS ZE PS c) r 0.5 M 1 np 10 nop 12 nc 10  nt 0 mode 'sugeno)
;l'ordre des règles n'est important que pour le mode exact d'inférence, les cas particuliers de début de partie étant à la fin

(set 'P0 '( (255(t (t ) (any)) (c (c) (any)) (c)) ; DD donnant-donnant confiant
           (246(t (t t) (any any))(t (t c)(any any)) (t (c t)(any any)) (c (c c) (any any)) (c (any) (any)) (c)) ; DDD
           (296(c (c) (any)) (t (t) (any)) (t)) ;DDM
           (264(t (any) (any)) (t)) ;méchant
           (293(c (any) (any)) (c)); gentil
           (270 (c (t) (any)) (t (c) (any)) (c)) ; CG
      (266 (c (any any) (t c)) (c (any any) (c t)) (t (any any) (c c)) (c (any) (c)) (c)) ; périodique gentille PG
      (291(t (any any) (t c)) (t (any any) (c t)) (c (any any) (t t)) (t (any) (t)) (t)) ; périodique méchante PM
      (261(t (t t t) (any any any)) (t (t t c) (any any any)) (t (t c t) (any any any)) ; majorité mou MM
           (t (c t t) (any any any)) (c (t c c) (any any any)) (c (c t c) (any any any))
           (c (c c t) (any any any)) (c (c c c) (any any any)) (t (t t) (any any)) (c (c c) (any any))
           (c (c t) (any any)) (c (t c) (any any)) (c (c) (any)) (t (t) (any))  (c)); majorité sur 3 coups
      (292 (t (t t t) (any any any)) (t (t t c) (any any any)) (t (t c t) (any any any)) ; majorité dur  MD
           (t (c t t) (any any any)) (c (t c c) (any any any)) (c (c t c) (any any any))
           (c (c c t) (any any any)) (c (c c c) (any any any)) (t (t t) (any any)) (c (c c) (any any))
           (t (c t) (any any)) (t (t c) (any any)) (c (c) (any)) (t (t) (any))  (t))))
(set 'P (valoriser (produc NP)))    (set 'p p0)
(set 'stat (list (list (caar p) (caar (last p))))) (initrans nop) (enchaine 500)

; TRANSITIONS les fonctionnelles sont sans argument, elles produisent une transition dont
; l'argument doit être un chromosome
(de creation () (list 'lambda '(C) (list 'append 'C (list 'list (list 'funcall "nouvregle M)))))
(de suppression () (list 'lambda '(C)
           (list 'if (list 'null 'C) 'C (list 'let (list (list 'p (list 'random 1 (list 'length 'C))) )
           (list 'append (list 'firstn 'p 'C) (list 'nthcdr (list '1+ 'p) 'C))))))
;Crossover1 est une famille d'opérateurs où un échange aléatoire de deux sous-suites est effectué
(de crossover1 () ; crossover C1 C2 retournera une liste dont les car et cdr seront les deux fils
     ; p indice du premier et q longueur de la sous-suite
    (list 'lambda '(C1 C2) (list 'let (list (list 'r (list 'min (list 'length 'C1) (list 'length 'C2)))
          (list 'p 0) (list 'q 0))
      (list 'setq 'p (list 'random 0 (list '1- 'r))) (list 'setq 'q (list 'random 1 (list '- 'r 'p)))
       (list 'cons (list 'regul (list 'append (list 'firstn 'p 'C1) (list 'firstn 'q (list 'nthcdr 'p 'C2))
                   (list 'nthcdr (list '+ 'p 'q) 'C1)))
          (list 'regul (list 'append (list 'firstn 'p 'C2) (list 'firstn 'q (list 'nthcdr 'p 'C1))  (list 'nthcdr (list '+ 'p 'q) 'C2)))))))

(de aplat (L) (cond ((null L) nil)
     ((atom (car L)) (cons (car L) (aplat (cdr L))))
     (t (append (aplat (car L)) (aplat (cdr L))))))
(de recons (R) (ifn (null R) (reconsbis (car R) (cdr R) (quotient (length R) 2))))
(de reconsbis (conc lp k) (mcons conc (firstn k lp) (list (nthcdr k lp))))
;Mutation est une famille d'opérations où un gène au hasard est remplacé par un nouveau puisé dans la liste G
(de comp (f g h) (list 'lambda '(x) (list f (list g (list h 'x))))) ; composée de 3 fonctions
(de transfo (L phi) ; retourne L où phi (à un argument) a été appliquée sur l'un des termes au premier
```

```
; niveau de L, choisi au hasard de façon uniforme
      (let ((p (random 0 (length L)))) (append (firstn p L) (cons (funcall phi (nth p L)) (nthcdr (1+ p) L)))))
(de transition (psi) (list 'lambda '(C) (list 'transfo 'C (list 'comp "recons
       (list 'lambda '(x) (list 'transfo 'x (list 'quote psi))) "aplat))))
(de succ (X) (succbis X PRED))
(de succbis (X L) (if (eq X (car (last L))) (car L) (cadr (member X L))))
(de pred (X) (succbis X (reverse PRED)))
(de mut (X) (hasard (ret X PRED)))
(de bruitage (X) (+ (sommet X) (gauss 0 sigma)))
(de mutation () (transition 'mut))
(de suivant () (transition 'succ))
(de precedent () (transition 'pred))
(de bruit () (transition 'bruitage))
;Suivant et Precedent sont des transitions remplaçant un gène au hasard par son suivant ou son précédent dans la liste
; des prédicats (mutation non uniforme)
; Transpo est une famille d'opérations où 2 gènes au hasard sont inversés
(de transpo ()  (list 'lambda '(C) (list 'transfo 'C (list 'comp "recons (list 'funcall "transpobis) "aplat))))
(de transpobis () (list 'lambda '(C)
       (list 'let (list (list 'p (list 'random 1 (list '1- (list 'length 'C)))) (list 'q 0))
       (list 'setq 'q (list 'random (list '1+ 'p) (list 'length 'C)))
        (list 'append (list 'firstn  (list '1- 'p) 'C); p q sont les rangs comptés à partir de 1
                      (list 'cons (list 'nth (list '1- 'q) 'C)  (list 'firstn (list '- 'q 'p 1) (list 'nthcdr 'p 'C)))
                     (list 'cons (list 'nth (list '1- 'p) 'C) (list 'nthcdr 'q 'C)))))))
;LES AUTRES
(de crossover2 () ; retourne sur C1 C2 deux croisements en profondeur (abandoné)
       (list 'lambda '(C1 C2) (list 'reconsresultat (list 'funcall (list 'crossover1) (list 'aplat 'C1)(list 'aplat 'C2)))))
(de retirp () (let ((i (random -9 0)))(list 'comp "recons  (funcall 'ajourandom i) "aplat )))
(de retirg () (let ((i (random -30 -19)))(list 'comp "recons  (funcall 'ajourandom i) "aplat )))
(de ajoup () (let ((i (random 0 9)))(list 'comp "recons  (funcall 'ajourandom i) "aplat )))
(de ajoug () (let ((i (random 20 30)))(list 'comp "recons  (funcall 'ajourandom i) "aplat )))
(de bruit () (let ((i (random -15 15)))(list 'comp "recons  (funcall 'ajourandom i) "aplat )))

(de ajourandom (i) ;produit une fonction ajoutant i sur un élément au hasard de son argument
       (list 'lambda '(C) (list 'let (list (list 'p (list 'random 0 (list 'length 'C))))
       (list 'append (list 'firstn 'p 'C) (list 'cons (list '+ (list 'nth 'p 'C) i) (list 'nthcdr (list '1+ 'p) 'C))))))
;Migration est une famille d'opérations où le quart environ des règles sont modifiées radicalement
(de migration () (list 'lambda '(C)
       (list 'let (list (list 'p (list 'random 6 (list 'length 'C))) (list 'q 0))
       (list 'setq 'q (list 'random 1 (list '- (list 'length 'C) 'p)))
       (list 'append (list 'firstn 'p 'C) (list 'mapcar "modif (list 'firstn 'q (list 'nthcdr 'p 'C))) (list 'nthcdr (list '+ 'p 'q) 'C)))))
; Transitions pour des chromosomes sous forme de listes plates de longueur fixe ng
;Copigauche est une famille d'opérations où un gène au hasard est recopié sur sa gauche (droite)
(de copig (n) (let ((p (random 2 (1- n)))) ; se généralise en translation à droite plus loin
       (list 'lambda '(C) (list 'append (list 'firstn (- p 2) 'C)  (list 'cons (list 'nth (1- p) 'C) (list 'nthcdr (1- p) 'C))))))
(de copid (n) (let ((p (random 1 (- n 2))))
           (list 'lambda '(C) (list 'append (list 'firstn (1- p) 'C)  (list 'mcons (list 'nth (1- p) 'C) (list 'nth (1- p) 'C)
           (list 'nthcdr (1+ p) 'C))))))
;Transuiv est une famille d'opérations où deux gènes consécutifs sont inversés
(de transuiv (n) (let ((p (random 1 n)))
       (list 'lambda '(C) (list 'append (list 'firstn (1- p) 'C)
           (list 'mcons (list 'nth p 'C) (list 'nth (1- p) 'C) (list 'nthcdr (1+ p) 'C))))))
; Transg est une famille d'opérations où une sous-suite (BED sur l'exemple) de longueur aléatoire est translatée sur sa ;gauche
avec recopie du dernier (ici D) ex: ACBEDCBAA --> ABEDDCBAA
(de transd  (n) (let (( p (random 1 n)) (q 1)); p indice du premier, q longueur de la sous-suite
       (set 'q (random 1 (1+ (- n p)))) (list 'lambda '(C) (list 'append (list 'firstn p 'C)
            (list 'firstn q (list 'nthcdr (1- p) 'C)) (list 'nthcdr (+ p q ) 'C)))))
(de transg (n) (let  ((p (random 2 (1+ n))) (q 1)); p indice du premier, q longueur de la sous-suite
            (set 'q (random 1 (- (+ 2 n) p))) (list 'lambda '(C) (list 'append (list 'firstn (- p 2) 'C)
                (list 'firstn q (list 'nthcdr (1- p) 'C)) (list 'nthcdr (- (+ p q) 2) 'C)))))
```

ERROR: undefined
OFFENDING COMMAND: ÿ

STACK: