Stratégies probabilistes à mémoire de 1 coup au dilemme itéré du prisonnier

Jean-Paul Delahaye, Philippe Mathieu

Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL (équipe SMAC)
Centre de Recherche en Informatique Signal et Automatique de Lille
F-59000 Lille, France
prenom.nom@univ-lille.fr

RÉSUMÉ. *Nous menons une étude expérimentale minutieuse sur les stratégies probabilistes au dilemme des prisonniers. Nous utilisons pour cela la méthode des classes complètes associée à une approche évolutionniste. Les résultats que nous obtenons ont donc un caractère objectif et dépendent le moins possible des ensembles de stratégies mis en compétition. Les ensembles étudiés sont grands (plusieurs milliers de stratégies), homogènes, et systématiques. Nous testons la robustesse de nos résultats par diverses méthodes. Les stratégies les meilleures repérées sont pour certaines d'entre elles nouvelles en ce sens qu'elles n'ont jamais été identifiées clairement par des études antérieures, et cela malgré leur simplicité. Nous identifions un critère jusque là inconnu qui conduit à une bonne anticipation du comportement des stratégies dans des univers variés. Nous confrontons les résultats de cette étude avec ceux obtenus par les approches mathématiques de Press et Dyson. Nous confrontons aussi les nouvelles stratégies avec les meilleures stratégies connues.*

ABSTRACT. *We conduct a thorough experimental study of probabilistic strategies to the prisoner's dilemma. To do this, we use the complete class method associated with an evolutionary approach. The results we obtain are therefore objective in nature and depend as little as possible on the sets of strategies put in competition. The studied sets are large (several thousand strategies), homogeneous, and systematic. We test the robustness of our results by various methods. The best strategies identified are for some of them new in the sense that they have never been clearly identified by previous studies, despite their simplicity. We propose a criterion that leads to a good anticipation of their behavior in various contexts. We compare the results of this study with those obtained by the mathematical approaches of Press and Dyson. We also confront the new strategies with the best known strategies.*

MOTS-CLÉS : *theorie des jeux, dilemme du prisonnier, stratégies mixtes, comportement.*

KEYWORDS: *game theory, iterated prisoner's dilemma, mixted strategies, behaviour.*

DOI:10.3166/RIA.32.141-167

1. Introduction

Suite à la parution de l'article (Press, Dyson, 2012) sur ce qu'ils ont appelé les stratégies ZD et l'extorsion, et suite aux réactions parfois critiques (Adami, Hintze, 2013 ; 2014 ; Hilbe, Nowak, Sigmund, 2013 ; Hilbe *et al*., 2014 ; Hilbe, Nowak, Traulsen, 2013 ; Liu *et al*., 2015 ; Milinski *et al*., 2016 ; Szolnoki, Perc, 2014a ; 2014b) concernant leurs conclusions, l'attention a été portée sur les stratégies probabilistes au dilemme itéré du prisonnier (Stewart, Plotkin, 2013 ; Delahaye, Mathieu, 2016 ; Dong *et al*., 2014 ; Hilbe, Nowak, Traulsen, 2013). Pourtant aucune méthode de tri systématique et aussi exhaustive que possible n'a été utilisée pour savoir si des stratégies probabilistes simples égalaient ou surpassaient les meilleures stratégies connues (Beaufils *et al*., 1997 ; Mathieu, Delahaye, 2015). Nous menons ici une telle étude en combinant deux des méthodes que nous considérons comme les plus susceptibles de produire des résultats robustes et dépourvus de subjectivité : la méthode des compétitions évolutionnaires parfois appelée *écologiques* (Axelrod, 2006 ; Beaufils *et al*., 1998 ; 1997 ; Delahaye, Mathieu, 2016 ; Sigmund, 2010), et la méthode des classes complètes (Beaufils, Mathieu, 2006 ; Delahaye, Mathieu, 2016 ; Mathieu *et al*., 1999). Nous utilisons en particulier des classes complètes homogènes composées de stratégies aléatoires.

Nos résultats montrent que des stratégies simples et pourtant inconnues jusqu'à présent émergent parmi les milliers de stratégies mises en compétition. Nous montrons que ces stratégies sont robustes et performantes pour les compétitions évolutionnaires, y compris mélangées à des stratégies non probabilistes reconnues. Un paramètre noté `p'` est identifié et interprété ; il est corrélé à la réussite des stratégies et semble donc fournir un critère efficace pour prévoir ce que donnera une stratégie probabiliste dans une compétition évolutionnaire. Des variantes plus complexes, mais plus fines de ce paramètre sont recherchées par une méthode d'exploration statistique exhaustive.

Des séries systématiques de tests sont menées pour s'assurer de la robustesse des résultats obtenus. En particulier nous plongeons les nouvelles stratégies probabilistes identifiées dans des environnements de stratégies déterministes pour nous assurer qu'elles restent performantes en dehors du contexte qui a permis de les découvrir. Les nouvelles venues sont aussi confrontées aux stratégies repérées par Press et Dyson et aux stratégies les meilleures connues pour le dilemme itéré du prisonnier. Cela nous conduit à une nouvelle formulation des conclusions expérimentales générales sur les stratégies optimales connues au dilemme du prisonnier.

2. Définitions et rappels

Le dilemme du prisonnier (Axelrod, 2006 ; Rapoport, Chammah, 1965 ; Sigmund, 2010 ; Kendall *et al*., 2007) est celui auquel sont soumis deux entités ayant le choix entre coopérer (`c`) ou trahir (`d` pour "Defect") et qui sont rétribuées par `R` points si chacune joue `c`, par `P` points si chacune joue `d`, et reçoivent respectivement `T` et `S` points si l'une joue `d` et l'autre `c`. On décrit ces règles avec les notations suivantes : `[c,c]`→`R+R`, `[d,d]`→`P+P`, `[d,c]`→`T+S`.

Pour que la situation soit celle d'un dilemme, on impose (Axelrod, 2006) : `T>R>P>S` et `T+S<2R`. On choisit le plus souvent les valeurs `T=5`, `R=3`, `P=1`, `S=0`.

Le nom "dilemme du prisonnier" provient d'un récit imaginaire. Deux suspects sont arrêtés armés devant une banque. Un juge essaie de les faire avouer séparément qu'ils s'apprêtaient à mener une attaque. Si l'un avoue — c'est-à-dire trahit son compère — et l'autre non — poursuit la coopération avec son compère —, `[d c]`, celui qui a avoué est libéré (rétribution de 5 années de liberté), celui qui n'a pas avoué va en prison 5 ans (rétribution nulle, il écope du maximum prévu pour une attaque de banque). Si les deux compères restent solidaires, `[c c]`, ils vont 2 ans en prison chacun pour port illégal d'arme (rétribution de 3 années de liberté par rapport aux 5 ans du pire cas). Si les deux bandits se trahissent simultanément, c'est-à-dire avouent, `[d d]`, ils vont chacun 4 ans en prison (rétribution d'une année de liberté par rapport au pire cas pour les remercier de leurs aveux). Dans une telle situation, la trahison est un comportement logique. Elle conduit toujours à un meilleur résultat que la coopération. En effet : (a) si l'autre entité coopère, j'obtiens 5 points en jouant `d` mais seulement 3 points en jouant `c` ; (b) si l'autre entité trahit, j'obtiens 1 point en jouant `d`, et 0 en jouant `c`. Il y a dilemme car collectivement les deux entités emportent 6 points en jouant `[c, c]` alors qu'elles en remportent moins en jouant `[c, d]` et encore moins en jouant `[d, d]`. L'intérêt collectif est que chacun joue `c`, mais une analyse logique individuelle conduit inévitablement à `[d, d]` qui est collectivement le pire cas !

Le dilemme est itéré quand on imagine que la situation de choix entre `c` et `d`, se présente périodiquement aux deux mêmes entités. Jouer consiste alors à choisir une stratégie qui, informée du passé (donc du comportement antérieur de l'adversaire et de son propre comportement) indique comment jouer le coup suivant.

La stratégie `tit_for_tat` (donnant-donnant) consiste à jouer `c` lors du premier coup, puis à jouer au coup n ce que l'adversaire a joué au coup $n-1$. La stratégie `per_ddc` joue `d` puis `d`, puis `c` et recommence indéfiniment la même séquence des trois coups. Lorsque `tit_for_tat` rencontre `per_ddc` la confrontation donne la suite de coups : `[c,d] [d,d] [d,c] [c,d] [d,d] [d,c] [c,d] [d,d] [d,c]` ... Cela rapporte donc en moyenne $\frac{(0+1+5)}{3} = 2$ points par coup à `tit_for_tat` et $\frac{(5+1+0)}{3} = 2$ points aussi par coup à `per_ddc`.

Certaines stratégies peuvent décider leurs coups au hasard. La stratégie `random` (lunatique) joue par exemple `c` dans 50 % des coups et `d` dans 50 % des coups en tirant au hasard équitablement. Un petit calcul montre que `random` opposée à `tit_for_tat` gagne en moyenne $\frac{9}{4} = 2,25$ points par coup, ce que gagne aussi son adversaire. Les stratégies aléatoires dans certaines variantes du dilemme ont été prouvées meilleures que les stratégies déterministes (Delahaye *et al.*, 2000), ce n'est pas le cas du dilemme itéré classique.

Un raisonnement élémentaire montre que (a) `donnant-donnant` ne perd jamais plus de 5 points, quel que soit son adversaire et la durée de la rencontre, et (b) qu'il ne fait jamais mieux que lui.

Dans un tournoi entre un ensemble varié de stratégies (chacune joue contre chaque autre, y compris elle-même, puis on totalise les points gagnés) Robert Axelrod a observé que `tit_for_tat` était une bonne stratégie. En confrontation un contre un, elle ne bat jamais personne, mais ne perd jamais plus de 5 points, quelle que soit la durée de la partie. Son comportement incite ses adversaires à la coopération, ce qui fait qu'elle obtient de bons scores dans les confrontations collectives, qui la placent dans le peloton de tête. Deux autres stratégies ont été identifiées comme tout aussi intéressantes : `pavlov` et `gradual` (voir annexe section 5).

– `pavlov` (Wedekind, Milinski, 1996): lors du premier coup, je coopère ; ensuite si au dernier coup joué, j'ai gagné 3 points ou plus je rejoue la même chose, sinon je change.

– `gradual` (Beaufils *et al.*, 1997) : je coopère au premier coup ; ensuite, lorsque mon adversaire me trahit, je le punis le coup suivant (comme `donnant-donnant` qui répond à une trahison au coup $n-1$ par une trahison au coup n), mais je suis plus sévère que `tit_for_tat` car je punis mon adversaire en jouant `d` pendant k coups consécutifs, où k est le nombre de trahisons passées de mon adversaire (mes punitions sont donc graduelles). Après une telle phase de rétorsion, je coopère deux fois de suite pour tenter de rétablir la paix.

Une multitude d'études (Beaufils, Mathieu, 2006 ; Beaufils *et al.*, 1997 ; Kendall *et al.*, 2007 ; Li *et al.*, 2011 ; O'Riordan *et al.*, 2000 ; Tzafestas, 2000), conduisent aux conclusions suivantes sur lesquelles un consensus est établi.

– Il n'y a pas de stratégie meilleure que toutes les autres, mais certaines sont mauvaises dans pratiquement tous des environnements possibles, alors que d'autres sont efficaces et réussissent bien (gagnent beaucoup de points) dans des tournois variés.

– Les bonnes stratégies sont les stratégies réactives (qui répondent quand on les trahit), qui prennent le risque de coopérer (elles commencent par coopérer et face à un adversaire qui coopère, elles ne tentent pas de trahir), et savent être indulgentes (après une trahison de l'adversaire elles finissent par pardonner pour renouer la coopération. C'est le cas de `gradual`.

2.1. Simulation de l'évolution

À côté de l'évaluation des stratégies obtenues en organisant des tournois variés, il existe des méthodes de tests simulant des évolutions auxquelles ne réussissent que les stratégies robustes.

Les "sélections évolutionnaires" sont l'une de ces méthodes. On met plusieurs exemplaires de chaque stratégie (par exemple 100) à tester dans une arène virtuelle et on y organise un tournoi. En fonction des points gagnés lors de ce tournoi, on fait évoluer les effectifs de chacune des stratégies ce qui définit une seconde génération. Les nouveaux effectifs sont par exemple proportionnels aux gains de chaque classe. La seconde génération produit selon la même méthode une autre génération, etc. Les stratégies gagnantes (celles dont les effectifs sont les plus importants) lors d'une telle évo-

lution sont bonnes dans des arènes variables, leurs bons classements possèdent donc une signification plus profonde que celui donné par un simple tournoi. Les stratégies gagnantes lors d'évolutions utilisant un grand ensemble de stratégies probabilistes sont, sauf exception, performantes aussi dans de nombreuses autres confrontations avec des stratégies déterministes : réussir dans un concours avec de nombreux participants assure sans surprise de réussir aussi dans de nombreux concours avec moins de participants !

Les résultats obtenus par ces calculs modélisent la sélection naturelle. Ils confirment souvent (mais pas toujours comme on va le voir) ceux des tournois et en accroissent les contrastes. Ils conduisent de plus à une conclusion surprenante : sauf dans de très rares cas, l'arène finit par n'être occupée que par des stratégies qui ne prennent jamais l'initiative de trahir (c'est le cas de `tit_for_tat` de `gradual` ou de `pavlov`). Au bout de quelques générations, l'arène est donc occupée par des stratégies qui ne jouent entre elles que des coups `[c,c]`. L'arène se trouve donc dans un état de coopération généralisée. Bien qu'il n'y ait aucune autorité de contrôle et que la tentation de la trahison soit présente pour tous à chaque coup joué, le jeu de l'évolution conduit presque toujours à l'élimination de toutes les stratégies qui succombent à cette tentation.

2.2. Les classes complètes

Pour mener des tests objectifs qui ne dépendent pas des stratégies repérées bonnes ou robustes, et pour se donner des chances de découvrir de nouvelles stratégies performantes, nous utilisons la *méthode des classes complètes* (Beaufils *et al.*, 1998 ; Delahaye, Mathieu, 2016 ; Mathieu, Delahaye, 2015) qui consiste à regrouper systématiquement toutes les stratégies ayant des capacités équivalentes ou fonctionnant selon un principe abstrait fixé. Nous considérons en particuliers les classes `Mem(X,Y)` qui regroupent toutes les stratégies dont le coup n dépend de manière déterministe des `X` derniers coups que la stratégie a joués et des `Y` derniers coups que l'adversaire a joués. Tant qu'il n'y a pas `max(X,Y)` coups joués, la stratégie utilise une amorce fixée une fois pour toutes. Une stratégie de `Mem(1,2)` se définit donc par les deux premiers coups qu'elle joue, puis par ce qu'elle fait quand le passé est par exemple `[d, dc]` (elle a joué `d` au coup $n-1$, et l'adversaire a joué `d` au coup $n-2$, et `c` au coup $n-1$. Dans ce cas il y a 8 passés possibles). Nous notons une telle stratégie par un nom du type `mem12_cdCDCCDDCC` avec la convention que la suite désigne les 2 premiers coups puis les répliques pour les 8 passés possibles pris dans l'ordre lexicographique des passés de soi-même suivi du passé de l'adversaire `[c cc] [c cd] [c dc] [c dd] [d cc] [d cd] [d dc] [d dd]`. Le nombre de stratégies de `Mem(1,2)` est $1024 = 2^{10}$ (il y a dix paramètres binaires à fixer pour chacune) et plus généralement le nombre de stratégies de `Mem(X,Y)` est $2^{max(X,Y)}.2^{2^{(X+Y)}}$.

2.3. Une sélection de 21 stratégies

Dans la suite de l'article nous utiliserons entre autres l'ensemble `Select` de 21 stratégies issues de (Mathieu, Delahaye, 2015) qui peut être considéré à la fois comme

contenant les stratégies les plus simples et les stratégies les meilleures identifiées aujourd'hui (voir section 5). On trouvera figure 1 le résultat de leur confrontation évolutionnaire. Cet ensemble noté est un jeu de test minimum : Trouver des stratégies qui se classent bien quand on les ajoute à `Select` est un défi difficile car cet ensemble regroupe toutes les stratégies les plus performantes actuellement connues.

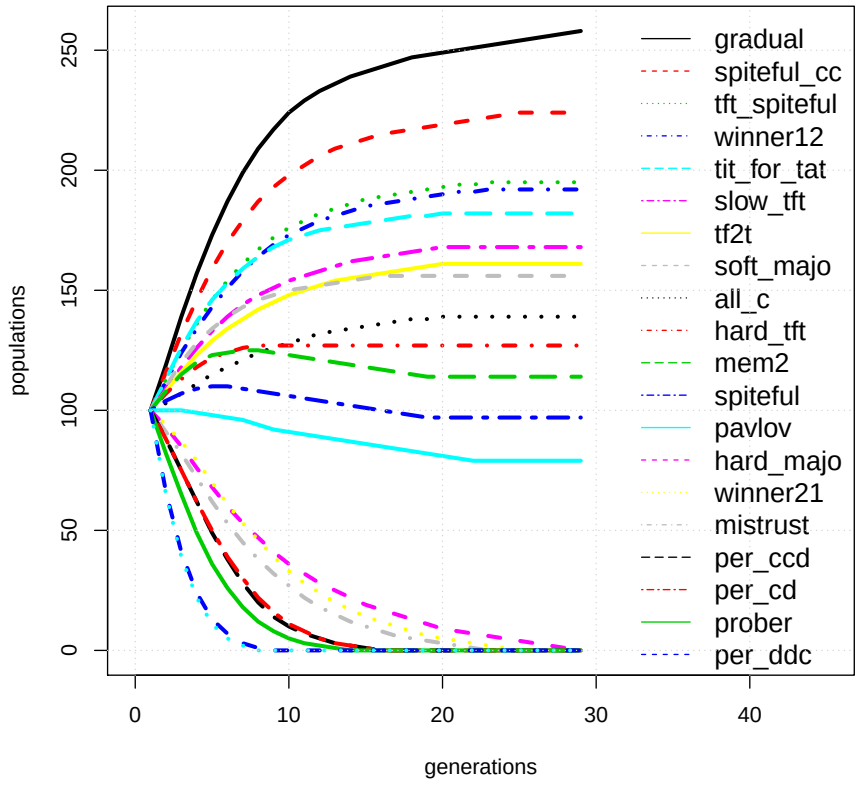

Figure 1. Compétition évolutionnaire des 21 stratégies contenues dans `Select`

3. Les résultats de Press et Dyson

3.1. Une percée théorique

L'article de William Press et Freeman Dyson (Press, Dyson, 2012) possède un titre provocateur : *Iterated Prisoner's Dilemma contains strategies that dominate any evolutionary opponent* ("Dans le dilemme itéré du prisonnier, il existe des stratégies

qui dominent tout adversaire évolutif"). L'observation quasi universelle d'une convergence vers la coopération généralisée semblait interdire qu'il existe des stratégies dominantes donc exploitant les autres. Les stratégies mises en avant par Press et Dyson ne respectent pas la règle qu'on pensait acquise : il ne faut jamais prendre l'initiative de trahir face à un adversaire qui coopère. Leur article (Press, Dyson, 2012) présente comme les meilleures possibles des stratégies simples (n'utilisant pour se déterminer que le dernier coup joué) ce qui est en contradiction avec l'idée d'un perfectionnement possible illimité des stratégies quand on accepte de les rendre plus complexes. Les affirmations proposées par (Press, Dyson, 2012) proviennent de raisonnements et de calculs mathématiques alors que dans ce domaine, il est difficile d'obtenir des résultats démontrés puisque l'espace associé au problème est infini et discret et n'est muni d'aucune topologie naturelle.

Une question simple se pose naturellement à laquelle les arguments mathématiques ne répondent pas : parmi un ensemble aussi neutre que possible de stratégies probabilistes soumis à un processus évolutif, quelles stratégies probabilistes s'imposent ? C'est la question que nous traitons ici.

Nos conclusions rejoignent d'autres conclusions déjà obtenues depuis la parution de l'article de Press et Dyson (Adami, Hintze, 2013 ; 2014 ; Hilbe, Nowak, Sigmund, 2013 ; Hilbe *et al*., 2014 ; Hilbe, Nowak, Traulsen, 2013 ; Liu *et al*., 2015 ; Milinski *et al*., 2016 ; Szolnoki, Perc, 2014a ; 2014b) mais grâce à notre méthode systématique, nous parvenons ici à mettre en avant une série de stratégies robustes et efficaces qui n'avait pas été extraites des résultats expérimentaux antérieurs et nous montrons qu'un critère simple permet de les identifier rapidement.

Un paramètre noté p' (et plusieurs de ses variantes) est identifié qui permet d'anticiper avec une bonne précision la réussite des stratégies probabilistes. Cette découverte positive tirée de nos expériences donne ce qui semble être le meilleur critère d'anticipation de la réussite dans les compétitions évolutionnistes entre stratégies probabilistes. Il apparaît indépendant des critères proposés par Press et Dyson.

Une seconde question est alors posée : ces stratégies probabilistes les meilleures sont-elles plus efficaces et robustes que les meilleures stratégies identifiées aujourd'hui ? Nous traitons cette question dans la dernière partie de l'article.

3.2. Stratégies probabilistes à mémoire d'un coup

Le travail de Press et Dyson propose deux théorèmes. Le premier théorème concerne le dilemme itéré dans une version limitée aux stratégies probabilistes à mémoire d'un coup : les stratégies `lunatique`, `tit_for_tat` et `pavlov` appartiennent à cette catégorie, mais pas la stratégie `gradual` qui pour décider ce qu'elle fait consulte le passé de tous les coups déjà joués.

Une stratégie à mémoire d'un coup est définie par 4 paramètres `p1`, `p2`, `p3`, `p4` qui indiquent la probabilité de jouer `c` lorsque le dernier coup a été `[c c]` ou `[c d]` ou `[d c]` ou `[d d]`. Notons `proba(p1,p2,p3,p4)` cette stratégie générale. On ne précise

pas comment est jouée le premier coup, mais c'est sans importance pour le résultat mathématique qui n'en dépend pas. En pratique pour les simulations, on considérera aussi bien les stratégies dont le premier coup est `c` que celles dont le premier coup est `d`.

La stratégie `tit_for_tat` est `proba(1, 0, 1, 0)` : elle coopère avec une probabilité de 100 % si le dernier coup est `[c c]` ou `[d c]` et coopère avec une probabilité de 0 % sinon. De même on vérifie sans peine que la stratégie `random` est `proba(`$\frac{1}{2}$`, `$\frac{1}{2}$`, `$\frac{1}{2}$`, `$\frac{1}{2}$`)` et que `pavlov` est `proba(1, 0, 0, 1)`. Press et Dyson considèrent une classe particulière des stratégies `proba(p1,p2,p3,p4)` dépendant de trois paramètres `a`, `b` et `c` et dénommées ZD. Nous les noterons `ZD(a,b,c)`.

3.3. Les stratégies ZD.

Les équations générales reliant les paramètres `p1`, `p2`, `p3`, `p4` pour les ZD dans le cas `R=3`, `S=0`, `T=5`, `P=1` sont :

`p1=1+3a+3b+c` `p3=5a+c`
`p2=1+5b+c` `p4=a+b+c`

Press et Dyson démontrent que lorsqu'on oppose une stratégie `ZD(a,b,c)` à une stratégie probabiliste à mémoire d'un coup `proba(p1,p2,p3,p4)`, et que l'on note `G1` le gain moyen (espérance) par coup de la première et `G2` le gain moyen (espérance) par coup de la seconde, alors ces gains moyens vérifient : `aG1+bG2+c=0`. Les deux stratégies ont des gains liés linéairement l'un à l'autre. Lorsqu'elles se rencontrent, `proba(p1,p2,p3,p4)` est en quelque sorte contrôlée par `ZD(a,b,c)`.

3.4. Les stratégies égaliseurs

Lorsque `a=0` et que `b`$\neq$`0` alors `G2=-c/b`, autrement dit une stratégie probabiliste quelconque a un gain moyen indépendant des probabilités qui la définissent, gain qui ne dépend que de la stratégie `ZD(a,b,c)` qui lui fait face. Une telle stratégie ZD est nommée égaliseur. Les relations deviennent :

`p1=3b+c+1` `p3=c`
`p2=5b+c+1` `p4=b+c`

et donc `G2=-c/b`.

Contre une telle stratégie, toutes les stratégies probabilistes à mémoire d'un coup obtiennent le même gain moyen qui est connu d'avance : `-c/b`. Inutile de se débattre face à une stratégie égaliseur vous gagnerez `-c/b` et pas plus ! Les valeurs possibles pour `-c/b` sont toutes les valeurs situées entre `P` et `R` , c'est-à-dire entre 1 et 3 quand on adopte les valeurs classiques de paramètres R, P, T et S.

Voici six exemples d'égaliseurs.

– EgaliseurA : a=0 b=-1/4 c=1/2 p1=3/4 p2=1/4 p3=1/2 p4=1/4 G2=-c/b=2

– EgaliseurB : a=0 b=-1/10 c=1/5 p1=9/10 p2=7/10 p3=1/5 p4=1/10 G2=-c/b=2

– EgaliseurC : a=0 b=-1/5 c=1/2 p1=9/10 p2=1/2 p3=1/2 p4=3/10 G2=-c/b=5/2=2,5

– EgaliseurD : a=0 b=-1/7 c=1/5 p1=27/35 p2=17/35 p3=1/5 p4=2/35 G2=-c/b=7/5=1,4

– EgaliseurE : a=0 b=-1/3 c=2/3 p1=2/3 p2=0 p3=2/3 p4=1/3 G2=-c/b=2

– EgaliseurF : a=0 b=-1/15 c=1/5 p1=1 p2=13/15 p3=1/5 p4=2/5 G2=-c/b=3

Voici les résultats des rencontres entre quelques stratégies connues et l'égaliseur `equa` qui est `ZD(0,-1/3,2/3)` ou encore `proba(2/3,0,2/3,1/3)`. On a donc `G2=-c/b=2`. Il force donc son adversaire à avoir un gain moyen de 2.

```
equa = 2,5 vs tit_for_tat = 2
equa = 3 vs gradual = 2
equa = 3 vs all_c = 2
equa = 1 vs all_d = 2
equa = 2 vs per-ccd = 2
equa = 1 vs spiteful = 2
equa = 2 vs equa = 2
```

Comme on le voit, `equa` force le gain moyen de l'adversaire, mais cela se fait parfois à ses dépens, et, par exemple contre `spiteful`, `equa` n'obtient qu'un point en moyenne par coup. Notons aussi que si une stratégie égaliseur force les stratégies rencontrées à avoir un faible score, elle en sera victime lorsqu'elle jouera contre elle-même !

3.5. Les stratégies extorqueurs

Parmi les stratégies ZD découvertes par Press et Dyson certaines opèrent une forme d'extorsion. En effet, si `c=-(a+b)P` (donc `a+b+c=0` avec `P=1`) on démontre que le gain moyen `G1` de la stratégie ZD contre une autre (obtenant le gain moyen de `G2`) vérifie `G1-P=X.(G2-P)` avec `X=-b/a`.

En clair, si la seconde veut gagner plus, donc augmenter `(G2-P)`, cela entraîne mécaniquement que la stratégie ZD augmente son gain moyen, dont l'écart à `P` est toujours `X` fois l'écart à `P` du gain moyen de la seconde.

Les quatre paramètres définissant ces stratégies extorqueurs sont donnés par les équations :

```
p1=2a+2b+1                    p3=4a-b
p2=4b-a+1                     p4=0
```

Voici quelques stratégies de cette catégorie.

– `ExtorqueurA : a=1/9 b=-1/6 p1=8/9 p2=2/9 p3=11/18 p4=0 X=-b/a=3/2`

– `ExtorqueurB : a=1/10 b=-1/5 p1=4/5 p2= 1/10 p3= 3/5 p4=0 X=-b/a=2`

– `ExtorqueurC : a=1/8 b=-1/6 p1=11/12 p2=5/24 p3=2/3 p4=0 X=-b/a=8/6`

– `ExtorqueurD : a=1/12 b=-1/6 p1=5/6 p2=1/4 p3=1/2 p4=0 X=-b/a=2`

– `ExtorqueurE : a=1/8 b=-1/5 p1=17/20 p2=3/40 p3=7/10 p4=0 X=-b/a=8/5=1,6`

– `ExtorqueurF : a=1/15 b=-1/5 p1=11/15 p2=2/15 p3=7/15 p4=0 X=-b/a=3`

Un défaut des stratégies de type extorqueur est que si `X>1` alors elles jouent mal contre elles-mêmes. Vouloir par exemple gagner deux fois plus contre son adversaire (par rapport à `P`) entraîne que face à elle-même elles ne gagneront que `P`, ce qui est moins bien que `C`.

Quand `X>1`, les extorqueurs ne vous permettent d'avoir un bon résultat que si vous leur en accordez un proportionnellement meilleur : `G1-P=X.(G2-P)`. Les extorqueurs sont donc des variantes de la stratégie `all_d` : personne ne peut les battre, mais cela a pour conséquence qu'elles prennent le risque de gagner peu. L'écart entre ce qu'elles gagnent en moyenne en plus de P (pris égal à 1 dans le cas usuel), est proportionnel à l'écart de ce que leur adversaire gagne en moyenne en plus de P. Si l'adversaire d'une stratégie extorqueur veut que cet écart soit grand pour lui, il le sera X fois plus pour l'extorqueur avec `X=-b/a`. Si vous jouez contre un extorqueur vous ne pouvez améliorer votre score moyen qu'en améliorant le sien: il vous taxe en proportion de vos revenus!

Prenons l'exemple de `ZD(1/10, -1/5, 1/10)` (`ExtorqueurB`) qui est la stratégie `proba(4/5, 1/10, 3/5, 0)`. C'est une stratégie `2-Extorqueur` : elle double à son profit l'écart à P de l'adversaire et par exemple quand elle joue contre `Pavlov` elle gagne en moyenne $1,62$ ($0,62$ point en plus de 1) alors que l'extorqueur obtient en moyenne $2,24$ ($1,24$ point en plus de 1). Face à elle, si vous devez choisir une stratégie à mémoire un coup, vous pouvez obtenir un gain plus ou moins grand, mais plus il sera grand, plus il le sera pour elle, et pour p points en moyenne au-dessus de $P = 1$, ce sera $2p$ points en plus pour elle au-dessus de $P = 1$. Les résultats mathématiquement démontrés sont bien sûr exacts.

On notera que les résultats concernant les stratégies égaliseurs avaient déjà été présentées dans (Boerlijst *et al.*, 1997) avant (Press, Dyson, 2012).

3.6. Reconnaître les ZD, les extorqueurs, les égaliseurs

En pratique connaissant `p1`, `p2` , `p3` , `p4` pour savoir si `proba(p1,p2,p3,p4)` est une stratégie de type ZD, il faut à partir de `p2 p3 p4` calculer :

$$\begin{aligned} a &= \frac{1}{15}p2 + \frac{4}{15}p3 - \frac{1}{3}p4 - \frac{1}{15} \\ b &= \frac{4}{15}p2 + \frac{1}{15}p3 - \frac{1}{3}p4 - \frac{4}{15} \\ c &= \frac{-1}{3}p2 - \frac{1}{3}p3 + \frac{5}{3}p4 + \frac{1}{3} \end{aligned}$$

et vérifier que : $p1 = 1 + 3a + 3b + c$

Si $a = 0$ et $b \neq 0$ cette ZD stratégie est alors un égaliseur. Si $a + b + c = 0$ cette ZD stratégie est alors un extorqueur.

3.7. Utilité d'une longue mémoire

Le second théorème important de l'article de Press et Dyson indique que si, dans une partie supposée infinie, la stratégie A est face à une stratégie B ayant une mémoire de k coups, il existe alors une stratégie A' qui obtient le même score moyen face à B et qui n'a qu'une mémoire de k coups. Autrement dit pour obtenir un certain résultat face à une stratégie donnée, il n'est jamais nécessaire d'avoir plus de mémoire qu'elle. La combinaison des deux résultats mathématiques de Press et Dyson conduit à l'affirmation que face à une stratégie égaliseur ou extorqueur ce ne sont pas seulement toutes les stratégies à mémoire un coup qui se trouvent contraintes mais toutes les stratégies à mémoire finie. De là on est tenté de conclure que : *"(a) les stratégies mémorisant plus que le dernier coup sont inutiles. (b) On dispose avec les stratégies ZD de stratégies dominantes dans l'absolu pour le dilemme itéré du prisonnier"*.

Certains ont d'ailleurs interprété ainsi les théorèmes démontrés, et le titre retenu pour leur article suggère que c'est le cas de Press et Dyson (bien que le contenu de l'article soit plus prudent). Pourtant la double affirmation sur l'inutilité des stratégies à mémoire étendue et sur la dominance absolue des stratégies ZD est fausse. Considérons d'abord l'affirmation de dominance des stratégies ZD. On sait depuis longtemps qu'au dilemme du prisonnier itéré battre son adversaire (obtenir plus de points que lui) peut se faire aux dépens de celui qui gagne et que celui-ci aurait pu obtenir plus de points en moyenne en acceptant d'être battu. La stratégie `all_d` (qui joue toujours `d`) bat tout autre stratégie (c'est une évidence) et par exemple sur une partie de 100 coups contre `tit_for_tat` obtient 104 points alors que `tit_for_tat` en gagne 99. La stratégie `all_d` bat `tit_for_tat`. La stratégie `all_c` (qui joue toujours `c`) ne bat pas `tit_for_tat` et gagne 300 points en 100 coups contre `tit_for_tat` qui gagne aussi 300 points. Contre `tit_for_tat`, `all_d` gagne peut-être, mais elle a tort de gagner car en faisant comme `all_c`, elle obtiendrait un bien meilleur score.

La plupart des stratégies ZD sont dans la même situation : elles ne gagnent contre leur adversaire qu'en renonçant elles-mêmes à avoir de bons scores. En un sens, ce que propose (Press, Dyson, 2012) est un ensemble de stratégies généralisant `all_d`. Les extorqueurs gagnent sur les stratégies auxquelles on les oppose, mais cela se fait aux dépens du total des points gagnés. D'ailleurs une stratégie extorqueur de paramètre X avec $X > 1$ qui joue contre elle-même n'obtient (d'après la théorie qu'on vérifie par simulation) qu'un seul point en moyenne par coup, ce qui est très médiocre. Il n'est pas vrai qu'être une bonne stratégie c'est battre son adversaire. Mieux vaut ne pas le battre, s'entendre bien avec lui et marquer beaucoup de points.

Le cas de `tit_for_tat` est d'ailleurs remarquable et tel qu'on a l'impression d'un paradoxe quand on énonce ses propriétés : `tit_for_tat` ne bat jamais aucune stratégie individuellement et se fait battre par de nombreuses stratégies, pourtant, c'est une bonne stratégie qui gagne de nombreux tournois et compétitions évolutionnaires. Elle gagne non pas parce qu'elle force les autres à gagner moins qu'elle, comme le fait une stratégie Extorqueur, mais parce qu'elle punit les stratégies qui ne veulent pas coopérer. Elle force la coopération. Face à elle, ou bien vous gagnerez peu de points, ou bien vous coopérerez, ce qui sera bon pour elle et pour vous.

C'est donc un contresens que croire que les stratégies ZD sont efficaces en terme de nombre de points gagnés. Dans des rencontres un contre un, elles dominent (comme `all_d`), mais pour cela elles se font mal, et globalement jouent plutôt mal. Outre l'erreur de croire que battre son adversaire c'est gagner des points, un autre oubli a conduit à croire que les stratégies ZD étaient dominantes : pour s'imposer il faut jouer correctement contre soi-même. C'est important dans les tournois, mais plus encore dans les compétitions évolutionnaires. En effet, si vous l'emportez lors des premières générations, l'arène se peuple de stratégies identiques à vous, et vous allez donc très fréquemment les rencontrer. Si vous jouez mal face à vous-même, cela se retourne à terme contre vous. Il se trouve que les stratégies ZD jouent mal contre elles-mêmes. Rien n'est faux dans les résultats mathématiques de Press et Dyson, mais en n'abordant que le problème *"qui gagne dans un combat un contre un ?"* et en oubliant le problème *"combien de points sont gagnés au total ?"* et le problème *"face à toi-même te fais-tu du tort ?"* les théorèmes démontrés ne permettent pas de conclure que les stratégies ZD sont de bonnes stratégies au sens évolutionnaire. Nos expériences de simulations montrent que les stratégies ZD sont mauvaises !

Le résultat de Press et Dyson sur l'inutilité pour une stratégie d'avoir de la mémoire est juste dans le sens précis suivant : Si A est face à une stratégie B ayant une mémoire de k coups, il existe une stratégie A' qui obtient le même score moyen face à B et qui n'a qu'une mémoire de k coups. Cependant cela ne signifie pas que face à deux stratégies B et C ayant une mémoire de k coups, il existe une stratégie A' à mémoire de k coups qui fasse le même score face à B et face à C. En effet, celle A' qui peut remplacer A face à B, n'est pas nécessairement la même que celle, A", qui face à C peut remplacer A. Pour affronter plusieurs adversaires avoir de la mémoire est utile tout simplement parce que cela permet de les distinguer les uns des autres.

Le résultat de Press et Dyson sur l'inutilité de la mémoire est valable uniquement dans les combats un contre un, mais cesse d'être vrai, dès qu'on envisage des tournois ou des compétitions évolutionnaires.

À nouveau les simulations confirment que les bonnes stratégies pour des environnements comportant plusieurs stratégies tirent un avantage de l'utilisation d'une large mémoire du passé. Sur les questions de la mémoire utile ou non on pourra consulter (Li, Kendall, 2014 ; O'Riordan *et al.*, 2000).

4. Stratégies probabilistes à mémoire de un coup

Dans une première série d'expériences nous considérons des ensembles aussi grands que possible et aussi homogènes que possible de stratégies probabilistes à mémoire d'un coup, et nous les soumettons à un processus évolutionnaire.

4.1. Expériences évolutionnistes massives

Pour obtenir des ensembles volumineux de stratégies probabilistes de la forme `proba(p1,p2,p3,p4)`, nous fixons un pas de variation des paramètres probabilistes. Pour `K=5` par exemple nous faisons varier les `pi` dans l'ensemble fini $0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1$ ce qui donne $2*6^4 = 2592$ stratégies (le 2 provient des deux choix possibles pour le coup initial). Nous notons cette classe complète `ProbaCD_K=5` dont on voit l'évolution figure 2.

Les résultats obtenus pour le tournoi sont indiqués ci-dessous. Nous avons utilisé des parties de 1000 coups et les paramètres usuels. Nous avons calculé les résultats de chaque partie un contre un en la faisant jouer 5 fois, cela de manière à limiter les effets des variations probabilistes.

```
1   probaD_0.0_0.0_0.0_0.2  39345809
2   probaC_0.0_0.0_0.0_0.2  39264669
3   probaD_0.2_0.0_0.0_0.2  39244163
4   probaD_0.0_0.2_0.0_0.2  39193409
5   probaC_0.2_0.0_0.0_0.2  39145486
6   probaD_0.4_0.0_0.0_0.2  39136783
7   probaC_0.0_0.2_0.0_0.2  39104825
8   probaD_0.2_0.2_0.0_0.2  39069164
9   probaC_0.4_0.0_0.0_0.2  39043576
10  probaD_0.6_0.0_0.0_0.2  38995942
```

La stratégie classée première est aussi notée `probaD(0,0,0,1/5)` ; le `D` indique que son premier coup est `D` (defect); l'entier en bout de ligne indique le gain en points lors du tournoi répété 5 fois.

La première stratégie ZD est 34^{me}, il s'agit de la stratégie équivalente à `spiteful`. Dans les 100 premières, il n'y a que six stratégies ZD. Sans surprise, les ZD, les extorqueurs et les égaliseurs ne réussissent pas particulièrement bien en tournoi. Ce constat

n'est pas étonnant puisque le critère qui a permis de les identifier ne prenait en compte que la capacité à faire mieux que l'adversaire dans des rencontres un contre un, dont on sait qu'elle n'est en rien garante d'une réussite en tournoi (il faut gagner beaucoup de points) ou en compétition évolutionnaire (il faut continuer à gagner beaucoup de points quand les stratégies inefficaces ont disparu).

Pour la compétition évolutionnaire, les dix premières stratégies avec leurs effectifs finaux sont données ici :

```
1   probaC_1.0_0.8_0.0_0.0  34262
2   probaC_1.0_0.6_0.0_0.0  31579
3   probaC_1.0_0.4_0.0_0.0  30550
4   probaC_1.0_0.2_0.0_0.0  28640
5   probaC_1.0_0.0_0.0_0.0  27746
6   probaC_1.0_0.0_0.0_0.2   9540
7   probaC_1.0_0.2_0.0_0.2   8893
8   probaC_1.0_0.0_0.2_0.0   8451
9   probaC_1.0_0.2_0.2_0.0   7701
10  probaC_1.0_0.4_0.2_0.0   5984
```

On notera que la stratégie classée cinquième correspond à `spiteful`

REMARQUE : Le mécanisme évolutif programmé consiste à remplacer chaque génération par une nouvelle génération dont les effectifs pour un type de stratégies donné sont proportionnels au nombre de points gagnés par les stratégies de ce type lors d'un tournoi général impliquant toutes les stratégies présentes à la génération précédente : la descendance d'un type de stratégies à la génération n est proportionnelle aux points gagnés par les stratégies de ce type dans ce tournoi entre stratégies de la génération $n-1$. Au départ, on considère que chaque effectif est 100 et on impose que l'effectif total d'une génération à l'autre reste le même (aux problèmes d'arrondis près). Nous avons aussi mené des tests quand l'effectif d'une stratégie à la génération n est obtenu en prenant a fois l'effectif de la stratégie à la génération $n-1$, et $(1-a)$ fois l'effectif donné par le calcul précédent, avec le paramètre a compris entre 0 et 1. Cela revient à considérer que le remplacement d'une génération par la suivante n'est que partiel : les parents ne meurent pas tout de suite ! Les résultats (classement et effectifs finaux) obtenus sont toujours très proches (si $a < 1$) de ceux qu'on obtient quand le remplacement d'une génération par une autre est total, seule la durée de convergence vers l'état de coopération généralisée s'allonge.

Comme c'est souvent le cas au dilemme, les résultats d'un tournoi dans une classe complète qui comporte beaucoup de stratégies médiocres ou mauvaises ne reflètent pas du tout ce qu'on trouve comme résultat de compétitions évolutionnaires. La raison en est simple : les stratégies qui profitent de la présence de mauvaises stratégies dans la soupe initiale, sont rapidement rétrogradées, voire éliminées lorsque les mauvaises disparaissent. Pour réussir dans un processus évolutif, il faut obtenir de bons résultats avec ceux qui obtiennent de bons résultats et qui sont les seuls sur le long terme à survivre. Le résultat d'un tournoi dans une soupe comportant beaucoup de stratégies médiocres n'a guère de sens, seul celui des compétitions évolutionnaires est pertinent.

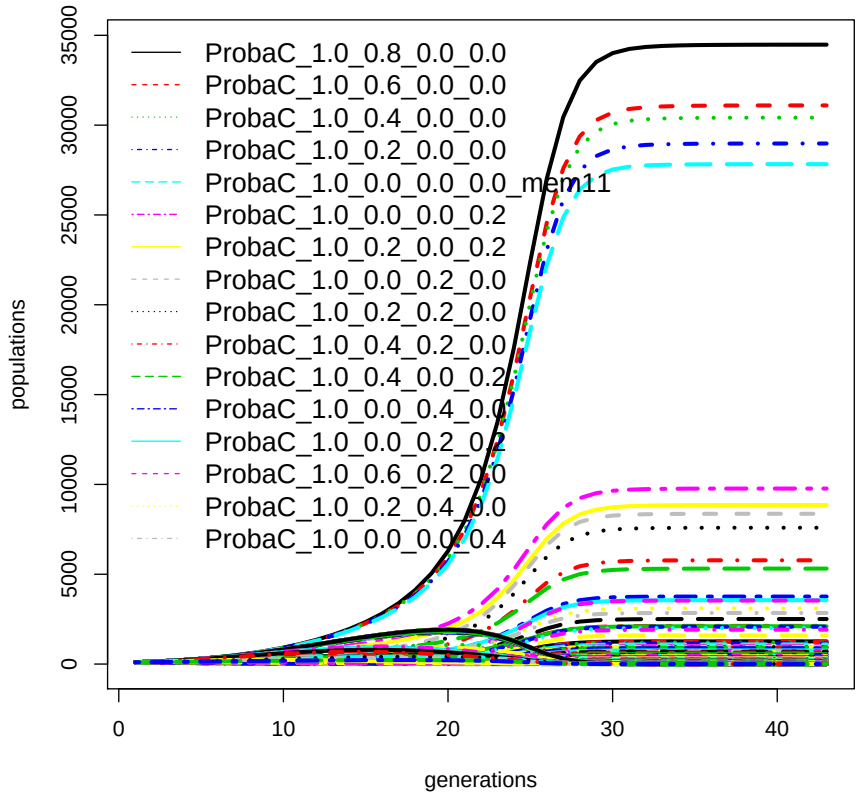

Figure 2. Compétition évolutionnaire des `ProbaCD_K=5`

La gagnante du tournoi est la variante de `all_d` consistant à remplacer le 4e paramètre `0.0` en `0.2` (`all_d` elle-même est classée 42^{me} de ce tournoi). Pour être bien classé au tournoi de la classe complète `probaCD_K=5`, il suffit d'exploiter les stratégies médiocres qui sont très nombreuses ; c'est très facile : il suffit de ne presque jamais coopérer. Cela ne nous apprend rien. Seule le résultat d'un processus évolutif qui commence par faire disparaître les stratégies médiocres présente de l'intérêt.

Dans le cas de la compétition évolutionnaire, ce qu'on découvre est remarquable et n'avait semble-t-il jamais été noté. La meilleure stratégie de notre large ensemble de plus de deux mille stratégies est : `probaC_1.0_0.8_0.0_0.0`. L'effectif initial de 100 est passé à 34262 quand l'état de coopération généralisé s'est installé.

Son comportement est d'une grande simplicité : c'est une rancunière (une fois qu'elle a commencé à trahir, elle trahit toujours puisque `p3` et `p4` valent 0), mais c'est

une rancunière qui réagit sans se précipiter quand on la trahit : en cas de coup `[c, d]`, elle ne se met à trahir qu'avec une probabilité de 20 %. Autrement dit, dans la phase initiale du jeu, elle coopère et continue de le faire aussi longtemps que l'autre coopère, et lorsqu'elle est trahie dans cette phase initiale, elle pardonne dans 80 % des cas. En revanche, lorsqu'elle se décide à trahir, il n'y a plus de retour possible à la coopération, elle trahit indéfiniment. Nous nommerons ce type des stratégies des *rancunières patientes*.

Les quatre stratégies suivantes sont elles aussi des rancunières patientes, mais dont la réactivité en cas de trahison augmente : d'abord 40 % pour la seconde, puis 60 % pour la troisième, puis 80 % pour la quatrième, puis 100 % pour la cinquième, ce qui donne la rancunière habituelle (`spiteful`).

```
probaC_1.0_0.6_0.0_0.0
probaC_1.0_0.4_0.0_0.0
probaC_1.0_0.2_0.0_0.0
probaC_1.0_0.0_0.0_0.0 (mem11)
```

On note sur les courbes que ces 5 stratégies sont largement en tête devant toutes les autres. La sixième stratégie est : `probaC_1.0_0.0_0.0_0.2`

La sixième stratégie possède encore un comportement qui s'interprète assez facilement. C'est une rancunière (sans aucune patience puisque `p2=0`), mais qui une fois dans sa phase de punition mène des tentatives de réconciliation : lorsque le coup qui vient d'être joué est `[d, d]`, elle coopère dans 20 % des cas, comme si elle disait à son adversaire : *nous sommes mal partis, je tente un premier pas (dans 20 % des cas) pour renouer une meilleure entente*.

Les suivantes sont, elles encore, susceptibles d'interprétations du même type, bien que de plus en plus compliquées. Nous proposons de nommer *rancunières conciliantes* toutes les stratégies de ce type. Pour être précis et mener des décomptes nous nommerons rancunières conciliantes toutes les stratégies `probaC(p1,p2,p3,p4)` avec `p1=1` et `p2+p3+p4` $\leq$ `1`. On notera que `tit-for-tat` et `Pavlov` sont de ce type. Les 37 premières stratégies dans la composition finale de la soupe après stabilisation de la compétition évolutionnaire appartiennent à cette catégorie.

Une autre remarque s'impose : dans la compétition évolutionnaire, seules 133 stratégies gardent des effectifs non nuls, et ce sont toutes des stratégies qui commencent par coopérer et qui vérifient `p1=1`. La soupe a donc incontestablement convergé vers un état de coopération généralisé.

4.1.1. Etude des ZD qui survivent

Les seules ZD qui survivent sont les suivantes. Elles sont indiquées avec leur classement et leur catégorie :

```
29   probaC_1.0_0.0_1.0_0.0_ZD_Extorq
34   probaC_1.0_0.6_0.4_0.0_ZD_Extorq
35   probaC_1.0_0.4_0.6_0.0_ZD_Extorq
72   probaC_1.0_0.2_1.0_0.4_ZD
125  probaC_1.0_0.6_0.6_0.4_ZD_Equa
```

La première ZD qui est donc 29^{me} est en fait `tit_for_tat` qui est effectivement une ZD de coefficient `X=1`. Ce `X=1` signifie qu'en réalité elle n'extorque rien, mais oblige ses adversaires à gagner autant qu'elle, ni plus, ni moins. Ce type de stratégies a parfois été appelée *generous extorquer* (Stewart, Plotkin, 2013) et leur capacité à survivre dans une compétition évolutionnaire a été identifiée bien supérieure à celles ayant un `X>1`. Ce que nous trouvons confirme bien que ce type d'extorqueurs a une certaine capacité à survivre, mais ce que nous observons est que ce ne sont ni les seules, ni les meilleures !

La seconde ZD qui est 34^{me} est une ZD avec `a=2/25`, `b=-2/25`, `c=0`, `X=1`. C'est une ZD de coefficient `X=1` (qui comme `tit_for_tat` n'extorque donc rien)

La troisième ZD qui est 35^{me} est une ZD avec `a=3/25`, `b=-3/25`, `c=0`, `X=1`. C'est encore une ZD de coefficient `X=1`.

La quatrième ZD, 72^{me} du classement, n'est ni extorqueur, ni égaliseur. C'est une ZD avec `a=2/25`, `b=-7/25`, `c=3/5`.

La relation liant le gain moyen qu'elle obtient `G1` et le gain moyen de son adversaire `G2` est : `aG1+bG2+c=0`. Ce qui donne : `2G1+15=7G2`. La stratégie dans la situation de coopération généralisée gagne 3 en moyenne par coup comme son adversaire.

La cinquième ZD, 125^{me} du classement, est de type égaliseur avec `a=0`, `b=-1/5`, `c=3/5`, `-c/b=3`. C'est une stratégie égaliseur qui oblige son adversaire à gagner 3 points en moyenne par coup, ce qui est le cas pour elle en cas de coopération généralisée.

4.2. Reconnaître les stratégies probabilistes efficaces ?

On le voit, les seules extorqueurs ou égaliseurs qui survivent sont en réalité des stratégies qui n'extorquent rien au sens strict puisqu'elles se contentent de gagner comme leur adversaire. Il est remarquable qu'elles soient largement battues par les rancunières patientes et des rancunières conciliantes qui sont donc dans ce contexte évolutif meilleures que la grande majorité des stratégies proposées par Press et Dyson.

Comme le montre la figure 3, le classement des 133 stratégies dont l'effectif ne s'annule pas est très directement corrélé au paramètre `p'=p2+p3+p4`.

Pour anticiper la réussite d'une stratégie probabiliste à un coup de mémoire, le double critère `p1=1` et `p'=p2+p3+p4` aussi petit que possible est très efficace.

Cela confirme les remarques faites sur le calcul mathématique de Press et Dyson. L'analyse menée par Press et Dyson en étudiant les résultats moyens de stratégies probabilistes dans des rencontres un à un, et en ne s'attachant qu'au fait de forcer et de contrôler l'adversaire sans se préoccuper du nombre de points que cela coûte, ne conduit à aucun critère d'identification des stratégies ayant une réelle efficacité dès qu'on se situe dans un contexte évolutif (ni même d'ailleurs dans le contexte de confrontations de type tournoi).

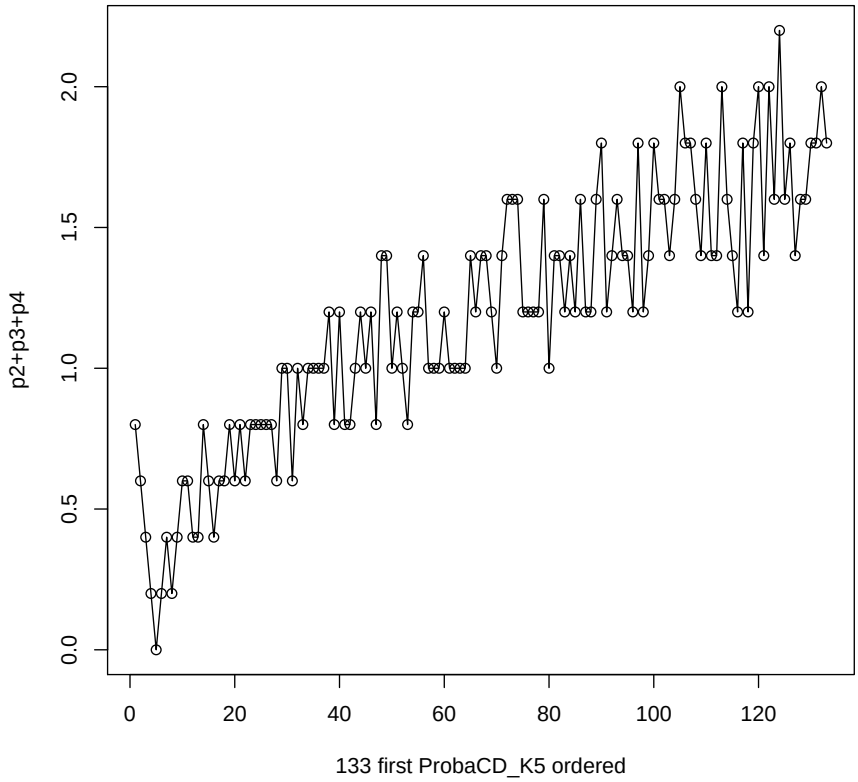

Figure 3. Evolution de p' en fonction du rang des stratégies de probaCD_K5. Plus la stratégie est performante, plus p' est faible

La figure 3 montre clairement que le paramètre `p'` est une bon prédicteur du rang d'une stratégie. Toutefois la croissance de `p'` est globale sans être parfaite et par exemple pour les 4 premières places, le paramètre `p'` va en décroissant. La raison est que le paramètre `p'` n'est sans doute pas assez précis pour permettre l'observation d'une croissance stricte en fonction du rang dans le classement. Il y a peut-être un ou plusieurs autres paramètres à faire intervenir pour créer un prédicteur parfait.

4.2.1. Comprendre le double critère

Nous proposons maintenant une interprétation et une explication de ce double critère. Pour gagner une compétition évolutionnaire ou seulement y réussir correctement, il faut survivre lorsque la coopération généralisée s'installe, il faut donc coopérer avec les stratégies qui coopèrent, d'où le `p1=1`. Il faut aussi être réactif, c'est-à-dire ne pas se laisser faire et adopter un comportement suffisamment sévère pour inciter l'autre à coopérer. La forme de réactivité la plus dure est celle de rancunière `p2=p3=p4=0`. Tempérer cette dureté est acceptable, si c'est modérément, et s'interprète de la manière suivante :

– choisir une valeur non nulle pas trop grande de `p2` revient à ne pas passer systématiquement dans l'état de rétorsion après un coup `[c, d]`, mais n'y passer qu'avec une certaine probabilité ;

– choisir une valeur non nulle pas trop grande de `p3` revient à accepter avec une certain probabilité après un coup `[d c]` de réessayer de coopérer avec un adversaire qui semble le désirer ;

– choisir une valeur non nulle pas trop grande de `p3` revient à accepter avec une certaine probabilité après un coup `[d d]` de faire le premier pas dans le but de faire renaître un état de coopération mutuelle.

Combiner ces trois formes de tempérance dans une stratégie en adoptant des petites valeurs non nulles de `p2` , `p3` et `p4` n'est pas absurde à condition que le total de la tempérance introduite dans son comportement ne soit pas trop grand, d'où le critère sur `p'=p2+p3+p4`.

Dans la figure 4 les stratégies de `ProbaCD_K=5` ont été groupées en 7 sous-ensembles dont on a calculé le classement moyen génération par génération. Il y a les stratégies pour lesquelles `p1`$\neq$`1`, puis pour celles avec `p1=1`, celles avec `p'` dans l'intervalle `[0;0,5[` puis `[0,5;1[`, `[1;1,5[`, `[1,5;2[`, `[2;2,5[`, `[2,5;3]`.

On voit que lorsque `p1`$\neq$`1`, au-delà de la génération 30 les classements deviennent médiocres, puis mauvais. Pour les stratégies avec `p1=1`, les meilleures valeurs possibles pour `p'` sont entre 0,5 et 1 (légèrement mieux que pour `p'` entre 0 et 0,5). Dès que `p'`$>$`1`, les classements sont nettement moins bons que pour `p'`$\leq$`1`.

4.2.2. Amélioration et variante de p'

Si nous calculons le coefficient de corrélation de Spearman entre le rang de classement des stratégies qui terminent avec des effectifs non nuls dans la compétition évolutionnaire `ProbaCD_K=5` (il y en a 133) et le paramètre `p'`, nous trouvons que `Cor(rang,p')=0,8805` ce qui est une très bonne corrélation. Nous avons cherché systématiquement à améliorer ce paramètre. Le paramètre suivant, encore très simple donne un remarquable résultat. `p"=p2+p3/2+p4` avec `Cor(rang,p")=0,9411231`. L'optimal, obtenu par analyse canonique des corrélations (CCA) permet encore une légère amélioration : `p* = 0,266 p2+0,138 p3+0,277 p4` avec `Cor(rang,p*)=0,9413768`. Les figures 3 et 4 illustrent cette corrélation.

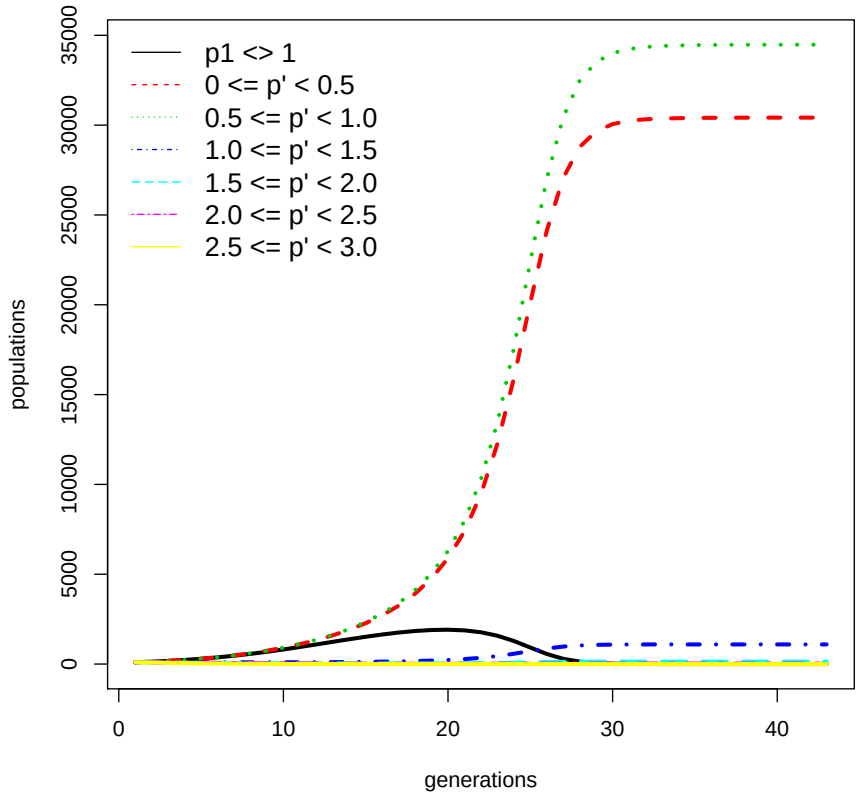

Figure 4. Synthèse par catégorie

4.3. Robustesse des expériences

Les résultats que nous venons de commenter et analyser sont-ils robustes ? Persistent-ils quand on change les paramètres précis de notre expérience avec 2592 stratégies. C'est ce que nous étudions maintenant.

Dans une autre expérience, nous n'avons mis au départ que des stratégies qui commencent par `c` (il y en a donc deux fois moins exactement), ce qui correspond à la classe complète que nous notons `ProbaC_K=5`) le résultat est très proche : les mêmes 5 premières se retrouvent avec quelques permutations du classement.

Avec `ProbaCD_K=4` nous obtenons des résultats équivalents : en tête les rancunières patientes classées par ordre de patience décroissante (`p2=75 %`, `p2=50 %`, `p2=25 %`, `p2=0`). Avec `ProbaCD_K=3` c'est encore la même chose.

Objection 1

Une objection pourrait être faite à notre méthode : les probabilistes composant la soupe initiale sont uniformément réparties (en faisant varier les `pi` par pas constant) : cette régularité pourrait entraîner des résultats spécifiques qui n'auraient donc pas de valeur générale.

Nous avons donc mené des expériences (voir un exemple sur la figure 5) où nous avons choisi 2000 (puis 4000) stratégies au hasard parmi la classe `ProbaCD_K=10` (qui comporte $2 * 11^4 = 29282$ stratégies).

Les résultats obtenus confirment l'expérience principale avec `ProbaCD_K=5` : les gagnantes sont à chaque fois des stratégies de type rancunière patiente ou rancunière conciliante. On y trouve la confirmation du double critère.

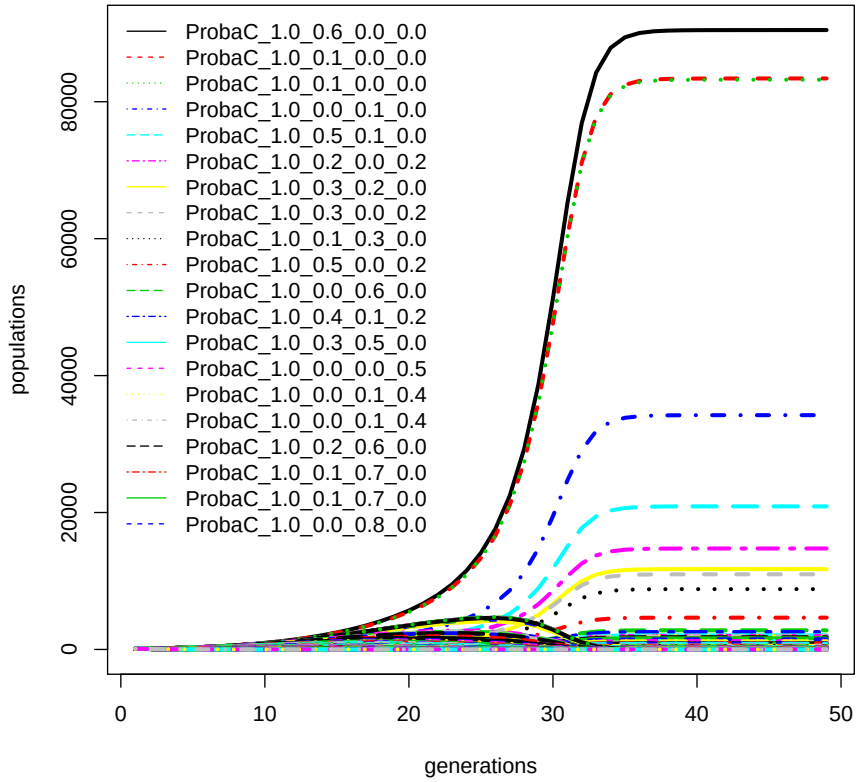

Figure 5. Un exemple de compétition réalisée entre 4000 stratégies prises au hasard parmi ProbaCD_K10

Voici les résultats et analyses de quelques expériences supplémentaires. Dans une expérience avec 2000 stratégies nous trouvons que seules survivent 5 stratégies :

```
1  probaC_1.0_0.1_0.3_0.0  112317
2  probaC_1.0_0.3_0.4_0.0   42330
3  probaC_1.0_0.3_0.4_0.0   39921
4  probaC_1.0_0.0_0.3_0.1    5077
5  probaC_1.0_0.3_0.3_0.1     354
```

Dans une autre il y a 11 survivantes :

```
1   probaC_1.0_0.0_0.0_0.1
2   probaC_1.0_0.4_0.0_0.1
3   probaC_1.0_1.1_0.1_0.0_ZD_Extorq
4   probaC_1.0_0.3_0.7_0.0_ZD_Extorq
5   probaC_1.0_0.1_1.1_0.0_ZD_Extorq
6   probaC_1.0_0.3_0.8_0.0
7   probaC_1.0_0.3_0.4_0.1
8   probaC_1.0_0.1_1.1_0.1
9   probaC_1.0_0.4_0.3_0.1
10  probaC_1.0_0.0_0.3_0.2
```

Les trois extorqueurs qui apparaissent en position 3, 4 et 5 ont toutes un coefficient X qui vaut 1 (ce ne sont donc pas au sens strict des extorqueurs), et surtout elles sont toutes du type rancunières patientes ou conciliantes.

Les 5 du premier calcul, et les 11 du second calcul possèdent toutes un $p' \leq 1$. On doit donc considérer qu'il est établi que pour réussir dans ce type de soupes, ce qui compte avant tout n'est pas d'être ZD, extorqueur ou égaliseur, mais de satisfaire au mieux le double critère (ou une de ses variantes avec p'' ou $p*$)

Objection 2

Une autre objection pourrait être formulée à notre méthode : nous ne considérons pas un assez grand nombre de ZD dans notre soupe initiale. Nous avons donc pris `ProbaCD_K=5` et ajouté une famille de 880 ZD (toutes celles dont les `pi` sont des multiples de $\frac{1}{32}$). Rien ne change pour l'essentiel : les 5 premières sont exactement les mêmes dans le même ordre que pour la soupe `ProbaCD_K=5` ; la première ZD est 25ème (à l'exception de rancunière qui est 5ème) et c'est `tit_for_tat` : `probaC_1.0_0.0_1.0_0.0_ZD_Extorq`.

La ZD suivante dans le classement est `ZD a=0.03125 b=-0.03125 c=0.0` dont le X vaut 1.

4.3.1. Robustesse des stratégies nouvelles

Nous avons voulu savoir si les meilleures stratégies déterministes connues telles que les conclusions de (Mathieu, Delahaye, 2015) les ont déterminées obtenaient de bons résultats dans ces classes complètes probabilistes.

Nous avons donc mené le calcul pour `ProbaCD_K=5` + `Select` .

Peu de choses changent concernant les positions relatives des ProbaCD mais les meilleures de `Select` s'intercalent et prennent de très bonnes places.

```
1   spiteful_cc              19286
2   tft_spiteful             19078
3   gradual                  18235
4   probaC_1.0_0.8_0.0_0.0   17944
5   probaC_1.0_0.6_0.0_0.0   16922
6   probaC_1.0_0.4_0.0_0.0   15759
7   probaC_1.0_0.2_0.0_0.0   14921
8   probaC_1.0_0.0_0.0_0.0   14147
9   mem2                     14109
10  spiteful                 14073
```

On note que la huitième correspond à `spiteful` version probabiliste. Les deux `spiteful` ne sont pas côte à côte à cause des fluctuations statistiques.

Toujours dans le but de tester la robustesse des stratégies identifiées, et en particulier pour voir ce qu'elles donnent quand elles se retrouvent parmi peu de probabilistes, nous avons composé une soupe avec les 20 premières de `ProbaCD_K=5`, les 1024 `Mem(1,2)` et les stratégies de `Select`. On obtient :

```
1   probaC_1.0_0.2_0.0_0.2   5255
2   tft_spiteful             4945
3   probaC_1.0_0.4_0.0_0.2   4877
4   probaC_1.0_0.2_0.0_0.4   4415
5   winner12                 4331
6   mem12_CCCDCDDCDD         4331
7   probaC_1.0_0.6_0.0_0.2   4081
8   mem12_CCCDCDDDDD         3557
9   spiteful_cc              3557
10  probaC_1.0_0.8_0.0_0.0   3551
```

La sixième du classement est en fait `winner12`, la huitième du classement est `spitefull_CC`. La onzième commence par CC et se fâche définitivement quand l'autre trahit deux fois de suite.

Les résultats (et de nombreux autres qui les confirment) sont très clairs : les stratégies les meilleures connues sont toujours bien classées (bien qu'elles proviennent d'expériences et de processus de sélection où seules des stratégies déterministes sont impliquées). Réciproquement, les nouvelles stratégies probabilistes repérées réussissent très bien dans des environnements composés presque exclusivement de stratégies déterministes (comme pour la dernière expérience).

5. Conclusion

L'expérimentation approfondie et systématique dans un modèle évolutionniste général et paramétré avec des stratégies de type "probabiliste à mémoire un coup" conduit à des résultats stables et que l'analyse permet d'anticiper une fois les bons paramètres

identifiés. La double condition que nous avons extraite et qui, semble-t-il, n'avait jamais été notée conduit à définir une classe particulière de stratégies probabilistes à mémoire d'un coup (les rancunières conciliantes) dont les membres se retrouvent systématiquement en tête de tous les classements des compétitions évolutionnaires que l'on peut imaginer, y compris quand on varie les soupes initiales pour y mettre de nombreuses stratégies extorqueurs, égaliseur ou ZD. De plus, les stratégies repérées se montrent robustes et efficaces dans des compétitions évolutionnaires composées de manière variées, par exemple ne contenant que des stratégies déterministes. De ce fait, elles rejoignent la famille des meilleures stratégies connues répertoriées dans (Mathieu, Delahaye, 2015). Les paramètres identifiés dans cet article ne sont néanmoins pas des prédicteurs parfaits et de nouvelles études sont sans doute à mener pour obtenir des conclusions encore plus précises que les nôtres. Une façon de les mener serait d'envisager les stratégies probabilistes ayant plus de mémoires des coups passés mais les prendre toutes avec des pas de variation des paramètres suffisamment petits n'est pas faisable aujourd'hui. D'autres méthodes sont à imaginer pour affronter la très grande complexité des interactions entre stratégies. À partir d'une définition élémentaire, le dilemme du prisonnier crée un problème d'une étonnante difficulté dont nous ne réussissons à comprendre les règles que progressivement ce qui illustre encore une fois que du simple peut naître des comportements et des dynamiques d'une richesse sans limite.

Bibliographie

Adami C., Hintze A. (2013). Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything. *Nature communications*, vol. 4.

Adami C., Hintze A. (2014). Corrigendum: Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything. *Nature communications*, vol. 5.

Axelrod R. M. (2006). *The evolution of cooperation*. Basic books.

Beaufils B., Delahaye J.-P., Mathieu P. (1997). Our meeting with gradual, a good strategy for the iterated prisoner's dilemma. In *Proceedings of the fifth international workshop on the synthesis and simulation of living systems, alife v*, p. 202–209. The MIT Press/Bradford Books.

Beaufils B., Delahaye J.-P., Mathieu P. (1998). Complete classes of strategies for the classical iterated prisoner's dilemma. In *International conference on evolutionary programming, ep7, vol1447*, p. 33–41. Springer.

Beaufils B., Mathieu P. (2006). Cheating is not playing: Methodological issues of computational game theory. *In Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, vol. 141, p. 185-189.

Boerlijst M. C., Nowak M. A., Sigmund K. (1997). Equal pay for all prisoners. *The American mathematical monthly*, vol. 104, n° 4, p. 303–305.

Delahaye J.-P., Mathieu P. (2016). Méta-stratégies pour le dilemme itéré du prisonnier. In *24e journées francophones sur les systèmes multi-agents (jfsma'16)*, p. 13–22.

Delahaye J.-P., Mathieu P., Beaufils B. (2000). The iterated lift dilemma. In *Computational conflicts*, p. 202–223. Springer.

Dong H., Zhi-Hai R., Tao Z. (2014). Zero-determinant strategy: An underway revolution in game theory. *Chinese Physics B*, vol. 23, n° 7, p. 078905.

Hilbe C., Nowak M. A., Sigmund K. (2013). Evolution of extortion in iterated prisoner's dilemma games. *Proceedings of the National Academy of Sciences*, vol. 110, n° 17, p. 6913–6918.

Hilbe C., Nowak M. A., Traulsen A. (2013). Adaptive dynamics of extortion and compliance. *PloS one*, vol. 8, n° 11, p. e77886.

Hilbe C., Röhl T., Milinski M. (2014). Extortion subdues human players but is finally punished in the prisoner's dilemma. *Nature communications*, vol. 5.

Kendall G., Yao X., Chong S. Y. (2007). *The iterated prisoners' dilemma: 20 years on*. World Scientific Publishing Co., Inc.

Li J., Hingston P., Kendall G. (2011). Engineering design of strategies for winning iterated prisoner's dilemma competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, n° 4, p. 348–360.

Li J., Kendall G. (2014). The effect of memory size on the evolutionary stability of strategies in iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, vol. 18, n° 6, p. 819–826.

Liu J., Li Y., Xu C., Hui P. (2015). Evolutionary behavior of generalized zero-determinant strategies in iterated prisoner's dilemma. *Physica A: Statistical Mechanics and its Applications*, vol. 430, p. 81–92.

Mathieu P., Beaufils B., Delahaye J.-P. (1999). Studies on dynamics in the classical iterated prisoner's dilemma with few strategies. In *European conference on artificial evolution*, p. 177–190.

Mathieu P., Delahaye J.-P. (2015). New winning strategies for the iterated prisoner's dilemma. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems*, p. 1665–1666.

Milinski M., Hilbe C., Semmann D., Sommerfeld R., Marotzke J. (2016). Humans choose representatives who enforce cooperation in social dilemmas through extortion. *Nature communications*, vol. 7.

O'Riordan C. *et al*. (2000). A forgiving strategy for the iterated prisoner's dilemma. *Journal of Artificial Societies and Social Simulation*, vol. 3, n° 4, p. 56–58.

Press W. H., Dyson F. J. (2012). Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, vol. 109, n° 26, p. 10409–10413.

Rapoport A., Chammah A. M. (1965). *Prisoner's dilemma: A study in conflict and cooperation* (vol. 165). University of Michigan press.

Sigmund K. (2010). *The calculus of selfishness*. Princeton University Press.

Stewart A. J., Plotkin J. B. (2013). From extortion to generosity, evolution in the iterated prisoner's dilemma. *Proceedings of the National Academy of Sciences*, vol. 110, n° 38, p. 15348–15353.

Szolnoki A., Perc M. (2014a). Defection and extortion as unexpected catalysts of unconditional cooperation in structured populations. *Scientific reports*, vol. 4.

Szolnoki A., Perc M. (2014b). Evolution of extortion in structured populations. *Physical Review E*, vol. 89, n° 2, p. 022804.

Tzafestas E. (2000). Toward adaptive cooperative behavior. In *Proceedings of the simulation of adaptive behavior conference, paris.*

Wedekind C., Milinski M. (1996). Human cooperation in the simultaneous and the alternating prisoner's dilemma: Pavlov versus generous tit-for-tat. *Proceedings of the National Academy of Sciences*, vol. 93, n° 7, p. 2686–2689.

Annexe

Liste des 21 stratégies constituant `Select`.

1. `all_c` (gentille) : je coopère toujours.

2. `all_d` (méchante) : je trahis toujours.

3. `tit_for_tat` (donnant-donnant) : je coopère au premier coup, puis je joue au coup n ce que l'autre a joué au coup $n-1$.

4. `spiteful` (rancunière) : je coopère au premier coup et tant que l'adversaire coopère, mais dès qu'il trahit, je trahis indéfiniment.

5. `soft_majo` (majorité-mou) : je coopère au premier coup et tant que mon adversaire a coopéré plus ou autant qu'il a trahi dans le passé ; sinon je trahis.

6. `hard_majo` (majorité-dur) : je trahis au premier coup et tant que mon adversaire a trahi plus ou autant qu'il a coopéré dans le passé ; sinon je coopère.

7. `per_ddc` (périodique-ddc) : je joue d, d, c, d, d, c, ...

8. `per_ccd` (périodique-ccd) : je joue c, c, d, c, c, d, ...

9. `mistrust` (méfiante) : je trahis au premier coup, puis je joue au coup n ce que l'autre a joué au coup $n-1$.

10. `per_cd` (périodique-cd) : je joue c, d, c, d, c, d, ...

11. `pavlov` : Je coopère au premier coup, puis je joue toujours coopérer sauf quand les deux joueurs n'ont pas joué la même chose au coup précédent.

12. `tf2t` (donnant-donnant-mou) : je coopère aux deux premiers coups, puis je coopère toujours au coup n, sauf si mon adversaire a trahi au coup $n-1$ et au coup $n-2$.

13. `hard_tft` (donnant-donnant-dur) : je coopère aux deux premiers coups, puis je coopère toujours au coup n, sauf si mon adversaire a trahi au coup $n-1$ ou au coup $n-2$.

14. `slow_tft` (donnant-donnant-lent) : je coopère aux deux premiers coups, ensuite je me mettrai à trahir lorsque l'autre aura trahi deux fois de suite, et je ne me remettrai à coopérer dès que l'autre aura coopéré deux fois de suite.

15. `gradual` (graduelle) : je coopère au premier coup et quand la règle suivante n'est pas en train de s'appliquer : à chaque fois que l'autre me trahit, je compte le nombre, n, de ses trahisons passées et je trahis n fois consécutivement suivi de deux coopérations.

16. `prober` (sondeur) : aux trois premiers coups, je joue trahir-coopérer-coopérer (d-c-d) ; ensuite, si aux coups 2 et 3 l'adversaire n'a pas trahi, je trahis toujours ; sinon je joue `tit_for_tat`.

17. `mem2` : je commence par jouer deux coups comme `tit_for_tat` ; ensuite je change de comportement pour deux coups en fonction des résultats des deux derniers coups, selon les règles : (A) si les deux coups précédents ont été `[c, c] [c, c]`, je passe à `tit_for_tat` ; (B) si le dernier coup a été `[c, d]` ou `[d, c]` alors je passe à `hard_tft` ; (C) dans les autres cas, je passe à `all_d`. De plus, si à un moment l'autre trahit deux fois de suite, je passe définitivement à `all_d` (Li, Kendall, 2014).

18. `winner12` (gagnante de `Mem(1,2)`) ; je coopère aux deux premiers coups puis je joue la table : `[C CC]->C [C CD]->D [C DC]->C [C DD]->D [D CC]->D [D CD]->C [D DC]->D [D DD]->D`

19. `winner21` (gagnante de `Mem(2,1)`) ; aux deux premiers coups je joue D C puis je joue la table : `[CC C]-> C [CC D]->D [CD C]->C [CD D]->D [DC C]->C [DC D->D [DD C]->D [DD D]->D`

20. `tft_spiteful` (donnant-donnant-rancunier) : je joue `tit_for_tat`, sauf que si on me trahit deux fois de suite, je me mets à trahir indéfiniment.

21. `spiteful_cc` (rancunièreCC) : je coopère aux deux premiers coups puis je joue `spiteful`.