Reconnaissance de Pattern dans des écosystèmes biologiques

# Introduction

Depuis quelques années, de nombreuses crises écologiques dues à l’impact de l’homme sur l’environnement se succèdent. Il est donc légitime de s’intéresser aux différents écosystèmes biologiques qui existent et à leurs fonctionnements, dans le but d’adapter notre rapport au monde qui nous entoure.

Les écosystèmes sont définis par de nombreux processus complexes, souvent de nature très différente. La dynamique de tels systèmes est difficile à analyser étant donné qu’elle résulte d’interactions entre de nombreux facteurs, tels que les différents être vivants, la composition du sol ou encore le climat. On cherche donc à mieux comprendre ces différentes interactions qui ont lieu dans ces écosystèmes.

Ainsi, dans ce TER, nous allons essayer de comparer les interactions de plusieurs écosystèmes, en essayant de trouver des points communs entre eux. Grâce à cela, comprendre un environnement pourrait nous aider à en comprendre un autre, nous permettant alors d’interagir avec celui-ci à des fins écologiques, sociales ou économiques.

# Définitions

Dans cette section, nous nous intéressons donc à la définition d’un écosystème, ainsi que de ces composantes.

## Ecosystème

Un écosystème est donc un ensemble d’agents qui peuvent être présents ou non.

Nous partons du principe qu’aucun agent ne peut être présent et absent à la fois.

La présence d’un agent, appelée polarité, est dénoté par a+ si l’agent a est présent, et a- sinon, celle-ci dépend des règles de réécritures R.

Plus formellement, un écosystème Ɛ est un tuple (A, R) tel que :

* A est un ensemble d’agents avec polarité tel que :

Ap = {a+, a- | a є A}

* R est un ensemble de règles de réécriture sous la forme :

r : α+, α- >> ω+, ω-

Où r est le nom de la règle, α+ et ω+ sont les agents présents et α- et ω-sont les agents absents.

On appellera lhs(r) (respectivement rhs(r)) les agents présents à gauche (respectivement à droite) d’une règle de réécriture r.

L’état d’un écosystème est défini par la présence ou non de tous ses agents. On le décrit donc par la liste de tous les agents présents, et on considère les autres comme absents.

La dynamique d’un écosystème (A, R) dépend de son état initial s0 et contient tous les états atteignables en appliquant les règles de R de façon non déterministe.  
Une règle r est applicable à un état s si les agents présents dans le côté gauche de la règle sont présents dans notre état s. En appliquant cette règle, on obtient alors un nouvel état s’ tel que :

s’= (s \ ω-) ᴜ ω+

## Exemple

Voici donc un exemple d’écosystème composé d’agents et de règles de réécriture :

**Agents :**

B : Oiseaux

I : Insectes

Pe : Pesticides

P : Pluie

**Règles :**

**** B+ I- Les oiseaux mangent les insectes

**** Pe-, R+ I+ La pluie et l’absence de pesticides laissent les insectes se développer

## Similarités entre deux écosystèmes

On s’intéresse donc dans cette section aux points communs repérables deux écosystèmes. Intuitivement, on pourrait exprimer ces similarités en comptant le nombre d’agents ayant des rôles identiques dans les deux écosystèmes.

Cela signifie que si on avait une carte de correspondance entre les agents de deux écosystèmes, et leurs règles, le « score » de similarité qu’on pourrait avoir serait défini par le nombre de correspondances entre agents et le nombre d’agents que deux règles auraient en communs.

Plus formellement, soit Ɛ1 = (A1, R1) et Ɛ2 = (A2, R2) deux écosystèmes, et µ et ρ les deux cartes de correspondances, respectivement entre agents et règles, la première est :

 µ : A1 A2

Cela signifie que l’agent a+ є A1 pourrait être associé ou à b+, ou à c- sachant que b+,c- є A2.

La seconde :

 ρ : R1 R2

suivant le même principe que pour µ mais représentant cette fois les correspondances entre les règles de nos deux écosystèmes Ɛ1 et Ɛ2.

Pour la résolution dans cplex, on utilisera deux matrices X et Y, correspondantes respectivement à nos cartes de correspondances µ et ρ.

Par exemple, si l’on avait deux écosystèmes :

**Ecosystème 1 :**

Agents : A, B

****Règles : A+ B+ A+ A-

**Ecosystème 2 :**

****Agents : C, D

Règles : C- D+ D- D+

La matrice X sera sous la forme :

A+ A- B+ B-

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

C+

C-

D+

D-

Ici, si la case [A+][C+] vaut 1, c’est que notre programme a trouvé que la comparaison entre nos environnement la plus optimale nous donne une correspondance entre les agents A+ et C+.

Un agent de notre premier écosystème ne peut pas être associé à plusieurs agents du second, la fonction de relation µ étant injective.

On a donc comme contrainte sur cette matrice :

-Il y a au plus un « 1 » par ligne



-Il y a au plus un « 1 » par colonne



Si par l’agent A+ correspond à l’agent C+, il faut également que A- soit jumelé à C-, on a donc, pour les agents, une contrainte de polarité consistante :

-ꓯa є A1, b є A2:

Xa+,b- = Xa-,b+ ʌ Xa+,b+ = Xa-,b-.

Concernant les correspondances entre règles de réécriture, la matrice Y sera :

****

**** A+ B+ A+ A-

|  |  |
| --- | --- |
|  |  |
|  |  |

**** C- D+

**** D- D+

****Si la première case vaut 1, c’est qu’il y a correspondance entre les règles

A+ B+ et C- D+.

La fonction ρ étant également injective, les seules contraintes à appliquer sur Y sont :

-Il y a au plus un « 1 » par ligne



-Il y a au plus un « 1 » par colonne



Pour que les résultats obtenus dans nos matrices X et Y nous renvoient les points communs entre nos deux écosystèmes, il faut maintenant mettre en place une fonction d’évaluation S.

## Fonction d’évaluation

La fonction d’évaluation, que nous nommerons S, est la contrainte qui permettra à notre modèle de « repérer » les ressemblances entre nos deux écosystèmes.

On lui attribuera comme valeur :



avec



et



Et on demandera à cplex de la maximiser.

La partie gauche est composée de



qui correspond aux nombres d’agents qui se correspondent entre les deux parties gauches, auquel on enlève deux malus :

Le premier :



est le nombre maximum d’agents qui pourraient être matchés, moins le nombre d’agents qui ont réellement été matchés.

Le second :



qui expriment le nombre d’agents qui ne pourront jamais être jumelé à cause de la différence de taille entre les côtés gauche des deux règles.

L’évaluation de la partie droite est identique.

# Mon travail

Le but a donc été pour moi dans un premier temps de créer des modèles cplex qui me permettraient d’obtenir en sortie les deux matrices de correspondances entre les agents et règles de deux écosystèmes, j’ai pour cela fait deux exemples, le premier assez simple, puis un second un peu plus complexe.

## Un premier exemple

Ce premier exemple est donc simplement composé de deux écosystèmes, ayant chacun deux agents et deux règles de réécriture.

Environnement 1 :

****Agents : A, B

Règles : A+ B+ A+ A-

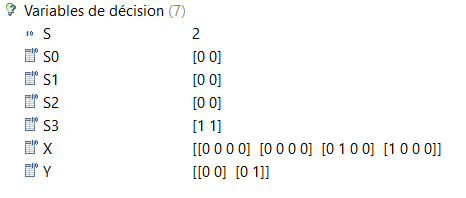
Environnement 2 :

****Agents : C, D

Règles : C- D+ D- D+

Pour comparer ces deux écosystèmes grâce à cplex, les contraintes à mettre en place correspondaient à celles vu dans la partie définitions, et ont donc été assez simples à écrire, à l’exception de notre fonction d’évaluation.

On obtiendra, en utilisant l’astuce vue dans la prochaine partie, comme résultat :



qui signifie

X :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A+ | A- | B+ | B- |
| C+ | 0 | 0 | 0 | 0 |
| C- | 0 | 0 | 0 | 0 |
| D+ | 0 | 1 | 0 | 0 |
| D- | 1 | 0 | 0 | 0 |
|  |  |  |  |  |

Y :

|  |  |  |
| --- | --- | --- |
|  | A+ >> B+ | A+ >> A- |
| C- >> D+ | 0 | 0 |
| D- >> D+ | 0 | 1 |

## Contraintes non linéaires dans la fonction d’évaluation

En effet, ici par exemple, la solution évidente serait de mettre en place la contrainte qui donnerait la valeur voulu à notre fonction d’évaluation :

*S == Y[0][0] \* (2\*X[1][0] - 1 + 2\*X[2][2] - 1) + Y[0][1] \* (2\*X[1][0] - 1 + 2\*X[2][1] - 1) + Y[1][0] \* (2\*X[3][0] - 1 + 2\*X[2][2] - 1) + Y[1][1] \* (2\*X[3][0] - 1 + 2\*X[2][1] - 1);*

Une des difficultés rencontrées a été la //multiplication de variables nécessaire au calcul de notre fonction d’évaluation. En effet cplex ne calcule pas les contraintes non linéaires. Pour pallier à ce problème, nous avons utilisé une astuce permettant indirectement d’avoir une contrainte contenant une multiplication de deux cases de tableaux dvar : sachant que nos valeurs sont des booléens, si Z==X\*Y n’est pas une contrainte calculable par Cplex, Z<=X, Z<=Y, Z>=X+Y-1 nous renverra le même résultat pour Z et sera accepté par cplex. Ainsi, en développant notre calcul de base, nous obtenons un résultat identique et juste.

Lors de la comparaison de ces deux environnements sur cplex, on obtient une correspondance entre des couples d’agents (A+, C-), (A-, C+), (B+, D+) et (B-, D-), et des règles (A+ ==> B+, C- ==> D+) et (A+ ==> A-, D- ==> D+).

Cette correspondance est bien celle qui nous donne le meilleur résultat pour notre fonction d’évaluation S, on se demande alors comment étendre notre modèle à des cas aux règles plus complexes, nous essayons donc un deuxième exemple :

## Un exemple plus complexe

Environnement 1 :

****Agents : A, B, C

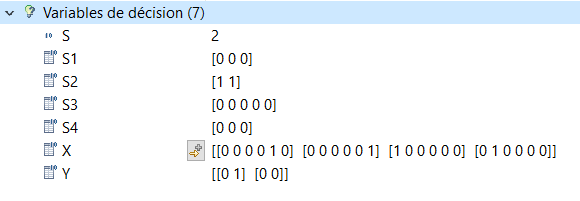
Règles : A+, B- B+ C- A+

Environnement 2 :

Agents : D, E

****Règles : D- E+ E+, E- E+

L’unique but de cet exemple a été de tester notre fonction d’évaluation sur des contraintes de différentes tailles, cela a correctement fonctionné, et nous a donné :



Qui peut être interprété :

X :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A+ | A- | B+ | B- | C+ | C- |
| D+ | 0 | 0 | 0 | 0 | 1 | 0 |
| D- | 0 | 0 | 0 | 0 | 0 | 1 |
| E+ | 1 | 0 | 0 | 0 | 0 | 0 |
| E- | 0 | 1 | 0 | 0 | 0 | 0 |

Y :

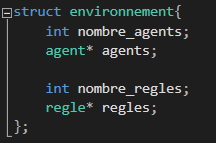
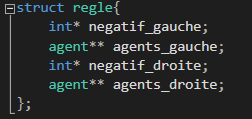
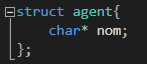
|  |  |  |
| --- | --- | --- |
|  | A+, B- >> B+ | C- >> A+ |
| D- >> E+ | 0 | 1 |
| E+, E- >> E+ | 0 | 0 |

## Notre programme

On cherche maintenant à créer un programme qui prendra en argument deux écosystèmes, et un fichier de sortie dans lequel il écrira le modèle cplex qui permettra leur comparaison.

J’ai choisi pour cela de coder en C, car c’est un langage polyvalent sur lequel je suis à l’aise, et qui permet facilement la lecture et l’écriture dans des fichiers.

Dans un premier temps, notre programme devra donc récupérer nos environnements et les stocker en mémoire, nous avons choisi pour cela de créer des structures :



Ainsi, la présence de pointeurs vers nos agents dans notre structure règle nous permettra de faire le calcul de notre fonction d’évaluation.

Le programme va alors lire les noms des écosystèmes mis en paramètre, s’ils sont bien formés, il les stockera dans nos structures.

Un fichier environnement bien formé est sous la forme :

*agents :*

*A+ :*

*B- :*

*rules :*

*A+, B+ >> A-*

Avec A et B deux agents quelconques.

Une fois nos deux écosystèmes récupérés, la fonction qui s’exécute ensuite va écrire dans un fichier du nom de notre troisième paramètre le modèle de comparaison cplex, composé donc dans l’ordre de :

-Déclaration de nos matrices X et Y, de nos variables intermédiaires pour le calcul des contraintes non linéaires dans la fonction d’évaluation, et de la fonction d’évaluation maximisé.

-Contraintes d’injectivités et de polarité consistantes pour les matrices X et Y.

-Contrainte pour le calcul des variables intermédiaires de notre fonction d’évaluation.

-Contrainte de calcul de la fonction d’évaluation

On obtient ainsi, à l’exécution du programme, un modèle bien formé, compilable et exécutable par cplex.

# Analyse des résultats

Une fois notre programme terminé, nous avons donc fait de nombreuses expériences pour en tester les limites et trouver des cas d’applications.

## Notre exemple

Le premier test que nous avons fait sur notre programme a été de reprendre le deuxième exemple

Environnement 1 :

Agents : A, B, C

Règles : A+, B- ==> B+ C- ==> A+

Environnement 2 :

Agents : D, E

Règles : D- ==> E+ E+, E- ==> E+

En y exécutant notre programme, le modèle cplex obtenu a bien été identique à celui que nous avions fait à la main, comme prévu.

## Le pattern proie prédateur

Le pattern proie prédateur est un ensemble de deux règles trouvables fréquemment dans différents écosystèmes.

Un test que nous avons réalisé a donc été de créer un écosystème contenant uniquement ce pattern :

**Agents :**

Proie, Prédateur

**Règles :**

Prédateur+ Proie-

Proie-Prédateur-

Nous l’avons ensuite comparé à d’autres écosystèmes, certains contenant ce pattern et d’autres non.

Par exemple, l’écosystème « mare.rr » :

**Agents :**

Eté, M, PP, PI

**Règles :**

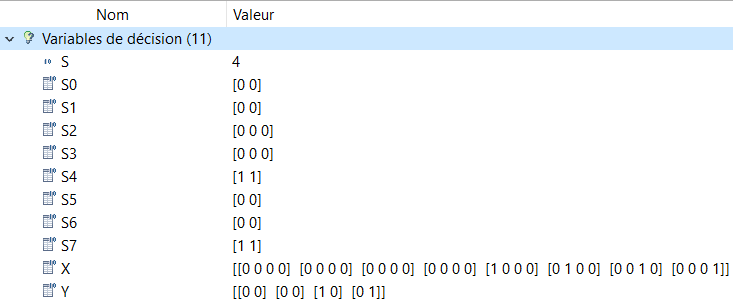
Eté+ >> M- # En été, la mare s'assèche

M+ >> PP+, PI+ # Quand il y a la mare, il y a les poissons

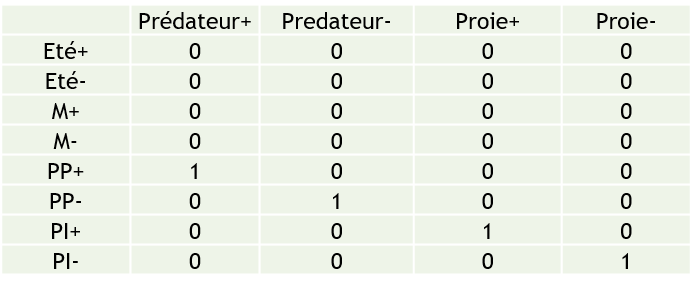
PP+ >> PI- # Le prédateur mange la proie

PI- >> PP- # Sans proie, le prédateur meurt

contient deux type de poissons, les poissons piscivores notés PP, qui se nourrissent d’autres poissons, et les poissons insectivore notés PI, qui se nourrissent d’insectes. Lorsqu’on compare ces deux écosystèmes, on obtient :



c’est-à-dire :

X :

Y :

|  |  |  |
| --- | --- | --- |
|  | Prédateur+ >> Proie- | Proie- >> Prédateur- |
| Eté+ >> M- | 0 | 0 |
| M+ >> PP+, PI+ | 0 | 0 |
| PP+ >> PI- | 1 | 0 |
| PI- >> PP- | 0 | 1 |

Ainsi, la comparaison de ces deux écosystèmes révèle bien une correspondance entre nos agents (Prédateur, PP) et (Proie, PI), et également une correspondance entre les règles :

(Prédateur+ Proie- , PP+ PI-) et

(Proie-Prédateur-, PI- PP-).

## Les limites

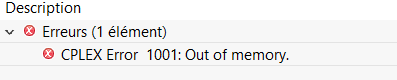
Nous avons ensuite essayé d’exécuter notre programme sur de plus grands écosystèmes :

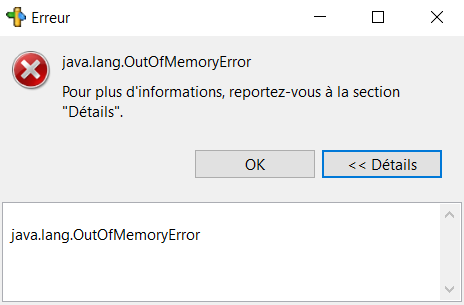
-Les Termites composés de 13 agents et 15 règles

-les Saisons composés de 46 agents et 97 règles

-la Camargue composé de 47 agents et 167 règles

Ainsi, l’exécution de notre programme sur les environnements termites et saisons nous donne un modèle d’environ 15000 lignes, mais cplex termine sur une erreur au bout d’un long temps de calcul :



Pour ce qui est de l’exécution sur Camargue et Saisons, le modèle engendré fait 260000 lignes, il est impossible de l’ouvrir à partir de cplex : 

Notre travail ne permet donc pas pour l’instant de comparer de grands écosystèmes, mais on peut toutefois rechercher de petits pattern tel que Proie-Predateur dans de grands écosystèmes.

# Conclusion

Ce travail de recherche nous a donc permis de faire des liens entre les différents écosystèmes, et de repérer des interactions types entre les agents qui les occupent. Cela nous permet donc de mieux comprendre leurs fonctionnements et facilite nos interactions avec ceux-ci à des fins écologiques.

Par la suite, certaines améliorations restent à apporter à notre programme, il serait en effet intéressant d’intégrer le solver cplex à notre code C, réduisant ainsi le nombre d’étapes avant l’obtention d’un résultat.

On pourrait ensuite implémenter une interprétation automatique de la sortie de cplex, dans le but d’avoir un affichage direct des correspondances entre agents et entre règles.

Enfin, on pourrait chercher des solutions pour résoudre les problèmes de dépassement mémoire qui ont lieu lors de l’exécution de trop gros modèle sur cplex, et ainsi comparer de plus grands écosystèmes.