

Reconnaissance de Pattern dans des écosystèmes biologiques

1. 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 les différentes interactions qui existent.

Si le principal intérêt de ce sujet est la compréhension de phénomènes biologiques, on peut également étendre le sujet à des systèmes plus physique voire sociaux-économiques.

Ainsi, dans ce TER, nous allons essayer de mettre en place une méthode automatique de comparaison d'écosystèmes à l'aide du solveur IBM ILOG cplex, notamment en recherchant des comportements types chez les agents qui y sont présents, ou en cherchant des similarités dans leurs interactions.

2. Définitions

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

Ecosystème

Un écosystème est un ensemble d'agents qui interagissent entre eux. Au départ, ceux-ci 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 \mathcal{E} est un tuple (A, R) tel que :

- A est un ensemble d'agents avec polarité tel que :

$$A^p = \{a^+, a^- \mid a \in A\}$$

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

$$r : \alpha^+, \alpha^- \gg \omega^+, \omega^-$$

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 $\text{lhs}(r)$ (respectivement $\text{rhs}(r)$) 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 s_0 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 :

$$s' = (s \setminus \omega^-) \cup \omega^+$$

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 :

$B+ \rightarrow I-$	Les oiseaux mangent les insectes
$Pe-, R+ \rightarrow I+$	La pluie et l'absence de pesticides laisse les insectes se développer

Similarités entre deux écosystèmes

On s'intéresse donc dans cette section aux points communs repérables entre 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 $\mathcal{E}_1 = (A_1, R_1)$ et $\mathcal{E}_2 = (A_2, R_2)$ deux écosystèmes, et μ et ρ les deux cartes de correspondances, respectivement entre agents et règles, la première est :

$$\mu : A_1 \rightarrow A_2$$

Cela signifie que l'agent $a+ \in A_1$ pourrait être associé ou à $b+$, ou à $c-$ sachant que $b+, c- \in A_2$.

La seconde :

$$\rho : R_1 \longrightarrow R_2$$

suivant le même principe que pour μ mais représentant cette fois les correspondances entre les règles de nos deux écosystèmes \mathcal{E}_1 et \mathcal{E}_2 .

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 : $A+ \longrightarrow B+$ $A+ \longrightarrow A-$

Ecosystème 2 :

Agents : C, D

Règles : $C- \longrightarrow D+$ $D- \longrightarrow D+$

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 environnements 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 est injective.

On a donc comme contrainte sur cette matrice :

-Il y a au plus un « 1 » par ligne

$$\forall m \in A_1 : \sum_{n \in A_2} X_{m,n} \leq 1$$

-Il y a au plus un « 1 » par colonne

$$\forall n \in A_2 : \sum_{m \in A_1} X_{m,n} \leq 1$$

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 :

$\forall a \in A_1, b \in A_2 :$

$$X_{a^+,b^-} = X_{a^-,b^+} \wedge X_{a^+,b^+} = X_{a^-,b^-}.$$

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

	$A^+ \rightarrow B^+$	$A^+ \rightarrow A^-$
$C^- \rightarrow D^+$		
$D^- \rightarrow D^+$		

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

$A^+ \rightarrow B^+$ et $C^- \rightarrow D^+$.

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

-Il y a au plus un « 1 » par ligne

$$\forall u \in R_1 : \sum_{v \in R_2} Y_{u,v} \leq 1$$

-Il y a au plus un « 1 » par colonne

$$\forall v \in R_2 : \sum_{u \in R_1} Y_{u,v} \leq 1$$

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 :

$$S(X, Y) = \sum_{u \in R_1, v \in R_2} Y_{u,v} (left(u, v) + right(u, v))$$

avec

$$left(u, v) = \sum_{\substack{n \in lhs(u), \\ m \in lhs(v)}} X_{n,m} - (\min(|lhs(u)|, |lhs(v)|) - \sum_{\substack{n \in lhs(u), \\ m \in lhs(v)}} X_{n,m}) - abs(|lhs(u)|, |lhs(v)|)$$

et

$$right(u, v) = \sum_{\substack{n \in lhs(u), \\ m \in lhs(v)}} X_{n,m} - (\min(|rhs(u)|, |rhs(v)|) - \sum_{\substack{n \in lhs(u), \\ m \in lhs(v)}} X_{n,m}) - abs(|rhs(u)|, |rhs(v)|)$$

Et on demandera à cplex de la maximiser.

La partie gauche est composée de

$$\sum_{\substack{n \in lhs(u), \\ m \in lhs(v)}} X_{n,m}$$

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

Le premier :

$$\min(|rhs(u)|, |rhs(v)|) - \sum_{\substack{n \in lhs(u), \\ m \in lhs(v)}} X_{n,m}$$

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

Le second :

$$abs(|rhs(u)|, |rhs(v)|)$$

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

L'évaluation de la partie droite est identique.

3. 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 : $A^+ \rightarrow B^+$ $A^+ \rightarrow A^-$

Environnement 2 :








Agents : C, D

Règles : $C^- \rightarrow D^+$ $D^- \rightarrow D^+$

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 comme résultat :

💡 Variables de décision (7)

 S	2
 S0	[0 0]
 S1	[0 0]
 S2	[0 0]
 S3	[1 1]
 X	[[0 0 0 0] [0 0 0 0] [0 1 0 0] [1 0 0 0]]
 Y	[[0 0] [0 1]]

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

Les agents A+ et D- ont été matchés, les règles A+ >> A- et D- >> D+ aussi. Les autres agents n'ont pas été matchés car ils ne se trouvent pas aux mêmes endroits dans les règles de réécritures, les matcher n'aurait donc eu aucun impact sur notre fonction d'évaluation S.

Contraintes non linéaires dans la fonction d'évaluation

La contrainte de calcul de notre fonction d'évaluation nous a posé problème. 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 :

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

Cependant, cplex nous renvoie un message d'erreur :

Description
<div> <div>Erreurs (1 élément)</div> <div> <div>CPLEX Error 5002: Q in 'q1' is not positive semi-definite.</div> </div> </div>

De ce que j'en ai compris, la multiplication de variable rend la résolution de la contrainte beaucoup plus complexe et celle-ci ne se fait plus en un temps linéaire, j'ai alors trouvé deux moyens de régler ce problème.

Pour le premier, j'utilise le fait que l'on travaille avec des booléens pour détourner le problème, en effet, si A, B et C sont des booléens, dire que

$$C = A * B$$

est équivalent à dire que

$$C \leq A \quad C \leq B \quad C \geq A + B - 1$$

J'ai donc utilisé des variables intermédiaires

```
dvar boolean S0[0..1];
dvar boolean S1[0..1];
dvar boolean S2[0..1];
dvar boolean S3[0..1];
```

et obtient alors un calcul de la fonction d'évaluation plus long, et composé de contraintes plus simples :

```
//Fonction d'évaluation
S0[0] <= Y[0][0]; S0[0] <= X[1][0]; S0[0] >= Y[0][0] + X[1][0] - 1;
S0[1] <= Y[0][0]; S0[1] <= X[2][2]; S0[1] >= Y[0][0] + X[2][2] - 1;

S1[0] <= Y[0][1]; S1[0] <= X[1][0]; S1[0] >= Y[0][1] + X[1][0] - 1;
S1[1] <= Y[0][1]; S1[1] <= X[2][1]; S1[1] >= Y[0][1] + X[2][1] - 1;

S2[0] <= Y[1][0]; S2[0] <= X[3][0]; S2[0] >= Y[1][0] + X[3][0] - 1;
S2[1] <= Y[1][0]; S2[1] <= X[2][2]; S2[1] >= Y[1][0] + X[2][2] - 1;

S3[0] <= Y[1][1]; S3[0] <= X[3][0]; S3[0] >= Y[1][1] + X[3][0] - 1;
S3[1] <= Y[1][1]; S3[1] <= X[2][1]; S3[1] >= Y[1][1] + X[2][1] - 1;

S == 2*S0[0] + 2*S0[1] - 2*Y[0][0]
    + 2*S1[0] + 2*S1[1] - 2*Y[0][1]
    + 2*S2[0] + 2*S2[0] - 2*Y[1][0]
    + 2*S3[0] + 2*S3[1] - 2*Y[1][1];
```

Le résultat obtenu est celui décrit précédemment et est bien optimal.

Pour régler ce problème, j'aurais également peut ajouter à mon code l'affixe « using CP ; ». En effet, cplex CP optimizer est un puissant outil qui permet notamment la linéarisation de certains problèmes d'optimisation tel que celui-là. J'ai malheureusement appris que cette commande pourrait m'aider trop tard, et ai donc continué en utilisant la première solution.

Un exemple plus complexe

L'unique but de cet exemple a été de tester notre fonction d'évaluation sur des contraintes de différentes tailles

Environnement 1 :

Agents : A, B, C

Règles : $A+, B- \rightarrow B+$ $C- \rightarrow A+$

Environnement 2 :

Agents : D, E

Règles : $D- \rightarrow E+$ $E+, E- \rightarrow E+$

Cela a correctement fonctionné, et nous a donné :

Variables de décision (7)	
S	2
S1	[0 0 0]
S2	[1 1]
S3	[0 0 0 0 0]
S4	[0 0 0]
X	[[0 0 0 0 1 0] [0 0 0 0 0 1] [1 0 0 0 0 0] [0 1 0 0 0 0]]
Y	[[0 1] [0 0]]

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 :

```
struct environnement{
    int nombre_agents;
    agent* agents;

    int nombre_regles;
    regle* regles;
};
```

```
struct agent{
    char* nom;
};
```

```
struct regle{
    int* negatif_gauche;
    agent** agents_gauche;
    int* negatif_droite;
    agent** agents_droite;
};
```

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.

4. 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 : $A+, B- \implies B+$ $C- \implies A+$

Environnement 2 :

Agents : D, E

Règles : $D- \implies E+$ $E+, E- \implies E+$

En y exécutant notre programme, le modèle cplex obtenu a été identique à celui que nous avons 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 :

Nom	Valeur
Variables de décision (11)	
S	4
S0	[0 0]
S1	[0 0]
S2	[0 0 0]
S3	[0 0 0]
S4	[1 1]
S5	[0 0]
S6	[0 0]
S7	[1 1]
X	[[0 0 0 0] [0 0 0 0] [0 0 0 0] [0 0 0 0] [1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]]
Y	[[0 0] [0 0] [1 0] [0 1]]

c'est-à-dire :

X :

	Prédateur+	Predateur-	Proie+	Proie-
Eté+	0	0	0	0
Eté-	0	0	0	0
M+	0	0	0	0
M-	0	0	0	0
PP+	1	0	0	0
PP-	0	1	0	0
PI+	0	0	1	0
PI-	0	0	0	1

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+ \longrightarrow Proie- , PP+ PI-) et

(Proie- \longrightarrow Prédateur-, PI- \longrightarrow PP-).

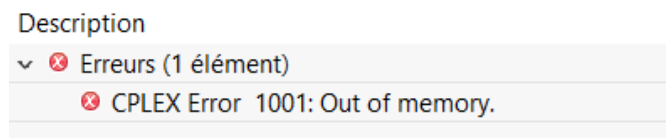
On a donc repéré ce pattern avec succès.

Les limites

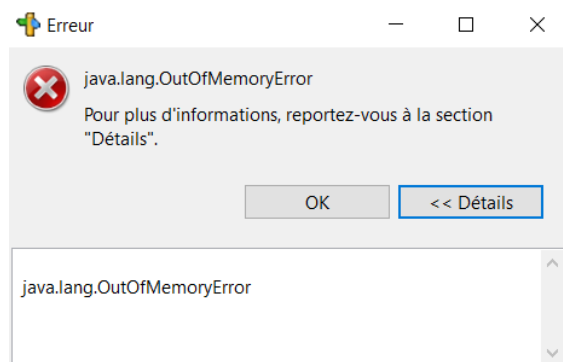
Nous avons ensuite essayé d'exécuter notre programme sur de plus grands écosystèmes :

- 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 avec 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.

5. 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 d'interagir avec ceux-ci en comprenant l'impact de nos actions, voire en les modifiant volontairement à des fins écologiques.

Par la suite, certaines améliorations restent à apporter à notre programme, il serait en effet intéressant d'intégrer le solveur cplex à notre code C, réduisant ainsi le nombre d'étapes avant l'obtention d'un résultat, ou d'utiliser l'outil cplex CP optimizer.

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.

Référence :

C. Di Giusto C. Gaucherel H. Klaudel F. Pommereau

« Pattern Matching in discrete Models for Ecosystem Ecology »