Modélisation et exploration d'un écosystème à dynamique spatiale par des outils de model-checking

Pierre Gaignon

▶ **To cite this version:**

Pierre Gaignon. Modélisation et exploration d'un écosystème à dynamique spatiale par des outils de model-checking . Statistiques [stat]. 2015. dumas-01253878

HAL Id: dumas-01253878

https://dumas.ccsd.cnrs.fr/dumas-01253878

Submitted on 11 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AGROCAMPUS OUEST

☐ CFR Angers

☑ CFR Rennes

Année universitaire : 2014.-2015

Spécialité :

Statistiques appliquées

Mémoire de Fin d'Études

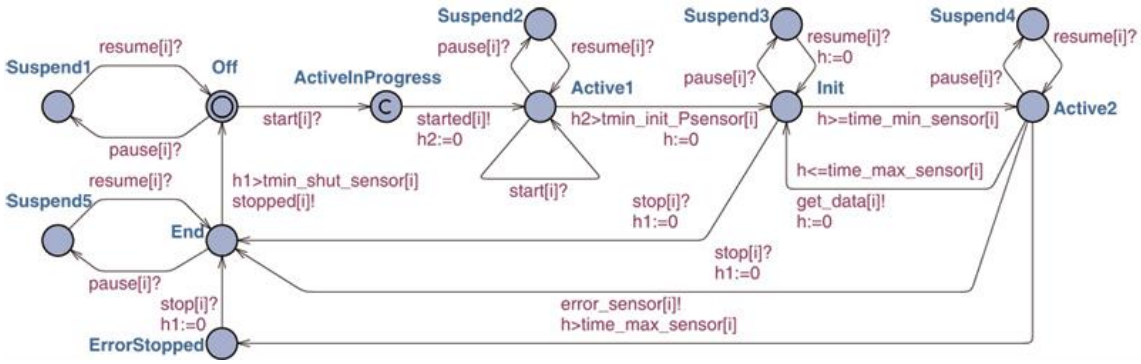
☑ d'Ingénieur de l'Institut Supérieur des Sciences agronomiques, agroalimentaires, horticoles et du paysage

☐ de Master de l'Institut Supérieur des Sciences agronomiques, agroalimentaires, horticoles et du paysage

☐ d'un autre établissement (étudiant arrivé en M2)

Modélisation et exploration d'un écosystème à dynamique spatiale par des outils de model-checking

Par : Pierre GAIGNON



Soutenu à Rennes, le 8 Septembre 2015

Devant le jury composé de :

Président :

Maître de stage : Christine LARGOUET

Enseignant référent : Julie JOSSE

Autres membres du jury (Nom, Qualité)

Nicolas PARISEY (co-encadrant)

Sylvain POGGI (co-encadrant)

Les analyses et les conclusions de ce travail d'étudiant n'engagent que la responsabilité de son auteur et non celle d'AGROCAMPUS OUEST

Fiche de confidentialité et de diffusion du mémoire

Confidentialité :

☑ Non ☐ Oui si oui : ☐ 1 an ☐ 5 ans ☐ 10 ans

Pendant toute la durée de confidentialité, aucune diffusion du mémoire n'est possible[(1)].
A la fin de la période de confidentialité, sa diffusion est soumise aux règles ci-dessous (droits d'auteur et autorisation de diffusion par l'enseignant).
Date et signature du maître de stage[(2)] :

Droits d'auteur :

L'auteur[(3)] autorise la diffusion de son travail

☑ Oui ☐ Non

Si oui, il autorise

☐ la diffusion papier du mémoire uniquement(4)

☐ la diffusion papier du mémoire et la diffusion électronique du résumé

☑ la diffusion papier et électronique du mémoire (joindre dans ce cas la fiche de conformité du mémoire numérique et le contrat de diffusion)

Date et signature de l'auteur :

Autorisation de diffusion par le responsable de spécialisation ou son représentant :

L'enseignant juge le mémoire de qualité suffisante pour être diffusé

☐ Oui ☐ Non

Si non, seul le titre du mémoire apparaîtra dans les bases de données.
Si oui, il autorise

☐ la diffusion papier du mémoire uniquement(4)

☐ la diffusion papier du mémoire et la diffusion électronique du résumé

☐ la diffusion papier et électronique du mémoire

Date et signature de l'enseignant :

(1) L'administration, les enseignants et les différents services de documentation d'AGROCAMPUS OUEST s'engagent à respecter cette confidentialité.
(2) Signature et cachet de l'organisme
(3).Auteur = étudiant qui réalise son mémoire de fin d'études
(4) La référence bibliographique (= Nom de l'auteur, titre du mémoire, année de soutenance, diplôme, spécialité et spécialisation/Option)) sera signalée dans les bases de données documentaires sans le résumé

Remerciements

Je tiens tout d'abord à remercier mes encadrants de stage, Christine Largouët de l'Agrocampus et de l'IRISA, et Nicolas Parisey et Sylvain Poggi pour l'équipe IGEPP de l'INRA du Rheu, pour m'avoir encadré pendant la durée de mon stage, pour avoir partagé leur savoir avec moi et pour m'avoir consacré du temps, que ce soit pour mon intégration au sein du laboratoire, la réalisation de ma mission mais aussi pour la recherche d'une thèse suite au stage.

Je remercie également toute l'équipe DREAM de l'IRISA qui m'a accueilli et intégré au cours de mon stage. Je voudrais notamment remercier Serge Emteu et Yann Dauxais, thésards au sein de l'équipe, pour m'avoir appris à utiliser des fichiers shell et l'accès à des serveurs de calculs dédiés pour mes simulations. Je voudrais aussi remercier Corentin Hardy et Clément Gautrais pour m'avoir aidé dans mes débuts avec Latex.

Enfin, j'ai une pensée toute particulière pour Anne-Isabelle Graux, chercheur à l'INRA au sein de l'UMR Pégase et membre de l'équipe DREAM, qui m'a été de précieux conseils dans mes recherches de thèse et m'a soutenu dans cette démarche.

Table des matières

Glossaire . . . . . A
Table des Figures C
Liste des tableaux C
Table des Annexes . . . . . E
Avant-propos . . . . . E

Introduction **1**

1 L'application et les enjeux **1**

2 La Modélisation qualitative **2**
2.1 Pourquoi un modèle qualitatif 2
2.1.1 Présentation des modèles qualitatifs 2
2.1.2 Les avantages et les inconvénients 3
2.1.3 Représentation de processus spatiaux 4
2.1.4 Représentation d'un écosystème marin : EcoMata 4
2.2 Les Automates temporisés 5
2.3 Model Checking et TCTL 7
2.4 Notion de coût en model-checking 8

3 Création des automates constitutifs du modèle **9**
3.1 Les Automates de Populations 10
3.2 Les Automates de Paysage 11
3.3 Les Automates de Migration et Diffusion 13
3.4 Les variables globales et le paramétrage du système 14

4 Contraintes liées aux automates **15**
4.1 Problème de taille de paysage 15
4.2 Limitation du nombre de transition 15

5 Sélection et Analyse d'un échantillon de paysage **16**
5.1 Sélection de Paysages 16
5.2 Génération des Paysages sélectionnés 17
5.3 Requêtes adressées aux modèles 17
5.4 Automatisation pour la vérification des requêtes 18
5.5 Analyse des résultats 19

6 Résultats **20**
6.1 Extraction des parangons 20
6.2 Résultats des Requêtes 22

7 Discussion **22**
7.1 Concernant la sélection de l'échantillon 22
7.2 Concernant les requêtes adressées 22

Conclusion **23**

Références **24**

Glossaire

A propos de la modélisation

model-checking : Le model checking désigne une famille de techniques de vérification automatique des systèmes dynamiques. Il s'agit de vérifier algorithmiquement si un modèle donné, le système lui-même ou une abstraction du système, satisfait une spécification, eventuellement formulée en termes de logique temporelle.

TA : Automate Temporisé, de *Timed Automata* en anglais. Il s'agit d'un formalisme de représentation de modèle qualitatif à=d'un système réactif en temps continu utilisé pour le model-checking.

TCTL : Timed Computation Tree Logic, logique temporelle avec des opérateurs quantitatifs temporels, notamment utilisée pour le model-checking.

A propos des automates temporisés

Localisation : Position d'un automate.

Etat : Combinaison de la localisation d'un automate et de son horloge.

Garde : Condition à vérifier pour permettre à un automate de changer de localisation.

Invariant : Condition à vérifier pour permettre à un automate de rester à une localisation donnée.

Transition : Permet l'évolution des automates, soit par une variation de localisation, soit par une incrémentation des horloges de manière synchrone.

Synchronisation : Communication entre plusieurs automates, avec un émetteur et un nombre de récepteurs variables (de zéro à tous les automates du système) leur permettant d'évoluer de manière simultanée. Cela permet de répandre une information rapidement. La ponctuation finale indique si l'automate est émetteur (point d'exclamation) ou récepteur (point d'interrogation). Il en existe deux types : chan et broadcast (voir définition ci-dessous).

broadcast : Forme de synchronisation, permet la communication entre plusieurs automates, cela nécessite uniquement un émetteur.

chan : Forme de synchronisation, permet la communication entre deux automates, cela nécessite obligatoirement un émetteur et un récepteur.

A propos des éléments agronomiques

adventices : Plantes qui poussent à un endroit où on ne souhaiterait pas qu'elles se développent.

nuisibles des cultures : Un organisme causant des dommages aux cultures.

carabes : Insectes de la famille des coléoptères qui sont souvent des prédateurs.

auxiliaire de culture : Être vivant qui détruit les ravageurs ou atténue leurs effets.

dispersion : Phénomène physique conduisant à la dissémination des proies à travers le paysage, mais nécessitant parfois un déclenchement extérieur, comme les récoltes. Elle peut se réaliser sur des parcelles non contiguës.

migration : Phénomène de déplacement de populations des prédateurs entre des éléments contigus d'un point de vue spatial

dynamique paysagère : Désigne les évolutions possibles d'un paysage par action humaine, c'est-à-dire les modifications des occupations du sol pour répondre à ses besoins.

élément (ou constituant) paysager : indique une partie du paysage (champ, prairie, portion de route, etc..)

occupation du sol : Nature des éléments présents sur un élément de paysage, que ce soit cultures, prairies, forêts mais aussi routes et zones urbaines. Il en existe 4 : champ cultivé, période inter-culturale (PIC), habitat semi-naturel(HSN) et matrice (route, habitation,etc..)

patch : Ensemble d'éléments géographiques de même nature ayant une contiguïté spatiale.

Table des figures

1 Interface d'EcoMata 4
2 Organisation simplifié d'un carrefour 5
3 Automate d'un feu de circulation 6
4 Automate d'un contrôleur de feux 7
5 Exemple d'automate temporisé avec coût 8
6 Modèle global désiré 9
7 Représentation des évolutions de population et interactions entre espèces 10
8 Automate de Population du Prédateur, exemple de la complexité des automates . . . 11
9 Représentation des éléments paysagers 12
10 Automate d'occupation du sol d'un élément du lattice 12
11 Représentation des communications entre les différents automates du modèle 14
12 Organisation pour l'automatisation de la vérification 19
13 Projection des variables sur les deux premières dimensions de l'ACP 21
14 Interface d'Uppaal pour les automates III

Liste des tableaux

1 Priorités de Migration chez les Prédateurs 13
2 Coordonnees des variables sur les 8 premieres dimensions de l'ACP XII
3 Valeurs propres et inertie sur les axes de l'ACP XIII
4 Cosinus carré des variables sur les huits premières dimensions de l'ACP XIII

Table des annexes

1 Définitions formelles des automates temporisés I
2 Utilisation d'Uppaal III
3 Paramétrage du modèle V
4 Fonction de gestion des rotations culturales VI
5 Règles pour les Migrations VIII
6 Ensemble des Synchronisations présentes dans le système IX
7 Définition des critères Paysagers X
8 Résultats de l'ACP XII

Avant-propos

Ce travail s'est déroulé au sein de l'IRISA à Rennes, en collaboration avec l'UMR IGEPP de l'INRA du Rheu. L'objectif du stage a été de proposer une nouvelle approche pour répondre aux problématiques des écosystèmes de taille importante. Il s'agit d'un élément d'une démarche plus importante visant à comparer deux méthodes, une qualitative, une quantitative. Ce rapport ne présente que la partie sur la création du modèle qualitatif et les résultats qui en découlent. Il ne présente à aucun moment la modélisation numérique ou une comparaison avec.

Pour cette première approche, de nombreuses simplifications ont été réalisées, notamment concernant les populations qui sont plus une illustration que de véritables composantes du modèle. Il a par exemple été fait abstraction des cycles de vie des différentes populations pour travailler sur des phénomènes plus continus dans le temps. Le modèle proposé s'intéresse plus à la représentation des déplacements de populations dans un paysage évoluant sous l'action humaine plutôt qu'à l'étude du développement même des populations.

Introduction

Actuellement, la modélisation de larges écosystèmes se confronte à la limite imposée par les calculateurs, à cause de l'explosion des combinaisons d'évolutions dans ces systèmes. La représentation de grands espaces avec une précision suffisante pose en effet le problème de la puissance nécessaire pour réaliser des simulations. C'est par exemple le cas pour les agro-écosystèmes, qui sont de plus en plus étudiés en raisons des services qu'ils rendent. De nouvelles approches se développent donc en parallèle pour pallier à ces problèmes. C'est notamment le cas avec des méthodes de modélisation qualitatives qui s'affranchissent d'un certain degré de précision pour omieux représenter entre les différents éléments constituants le modèle. C'est dans ce cadre que s'inscrit la collaboration entre l'IRISA et l'équipe IGEPP de l'INRA.

L'objectif fixé était une modélisation qualitative d'un écosystème à dynamique spatiale, situé dans un paysage agricole. Nous nous intéressons au rôle que peut jouer ce paysage sur un couple proie/prédateur avec sa possibilité de transformation, c'est-a-dire avec ses évolutions sous l'action humaine. Nous proposons donc une nouvelle approche, étant donné que rien n'a encore été réalisé sur la modélisation spatiale de populations d'un point de vue qualitatif. La modélisation de population par des automates a déjà été réalisée, mais rien en ce qui concerne le déplacement d'individus entre différentes zones géographiques. Nous prendrons deux espèces comme exemples pour le couple proie/prédateur, même si leur représentation n'est pas l'enjeu de cette modélisation, et que leur présence a plutôt pour but d'illustrer notre propos. L'objectif est donc de tester un large panel d'évolutions de notre paysage initial, pour étudier les facteurs paysages qui influeraient sur des critères agronomiques et écologiques.

Le problème dans le cas présent est la représentation de processus spatiaux dans un paysage agricole où existent deux espèces d'un couple proie/prédateur et connaître le rôle du paysage sur ces deux espèces par des outils de modélisation qualitative, à savoir des automates temporisés.

Pour cela, nous présenterons le domaine d'application et les enjeux qui y sont reliés, avant de voir les caractéristiques de la modélisation qualitative, et plus précisément celles des automates temporisés, afin de voir le modèle qui a été créé avec. Ensuite, nous parlerons des contraintes que nous avons sur les paysages, puis comment nous réalisons une sélection de paysage à tester avant de voir les résultats et d'en discuter.

1 L'application et les enjeux

L'objectif de ce stage étant avant tout de proposer une nouvelle approche aux méthodes classiques de modélisation d'écosystème, l'exemple est plus à objectif illustratif que prédictif. Les représentations des espèces, les adventices et les carabes, ont par conséquent été simplifiées. Par exemple, les cycles de vie des deux espèces n'ont pas été représentés et mais remplacés par des évolutions indépendantes des saisons pour alléger cette première approche. Les deux espèces exemples ont été choisies notamment car leur méthode de déplacement au sein de paysage est différente, de la dispersion pour les adventices, et de la migration pour les prédateurs.
Deux d'objectifs peuvent être fixées quand on étudie les écosystèmes dans les paysages agricoles : des enjeux agricoles et des enjeux écologiques, qui dépendent des cultures et du paysage. Dans notre cas, nous nous intéressons à la propagation des graines d'adventices, des mauvaises herbes, au sein d'un paysage agricole.

1. D'un point de vue agricole, nous chercherons à diminuer la quantité d'adventices présentes sur les cultures.
2. D'un point de vue écologique, l'objectif sera de diminuer l'utilisation des pesticides.

Un des moyens recherché est de trouver une organisation du paysage défavorable au développement des adventices, ou favorable à celui des carabes.
L'objectif étant d'étudier les évolutions possibles d'un paysage existant, ce qu'on appellera des dynamiques paysagères, le travail se décomposera en quatre étapes :

1. Création d'un modèle qualitatif.
2. Générer l'ensemble des paysages et sélectionner ceux que l'on désire tester.
3. Explorer les dynamiques sélectionnées.
4. Analyser les résultats.

2 La Modélisation qualitative

Dans des modèles de modélisation classique, le point de départ de la représentation d'un écosystème avec un couple de type proie/prédateur est l'utilisation des équations de Lotka-Volterra [13].

$$\frac{dx(t)}{dt} = x(t)(\alpha - \beta y(t)) \tag{1}$$

$$\frac{dy(t)}{dt} = -y(y)(\gamma - \delta x(t)) \tag{2}$$

$x(t)$ représente la quantité de proie présente à l'instant t, alors que $y(t)$ représente les prédateurs. Les deux équations (1) et (2) différentielles ne peuvent être traitées séparément, et leur résolution doit être simultanée. Il n'existe cependant pas de résolution formelle. Les méthodes usuelles utilisent des résultats approchés, soit à partir d'hypothèses simplifiées, soit par des résolutions par étapes. Il est donc difficile de modéliser ces deux espèces de façon rigoureuse.

Nous allons donc proposer une nouvelle approche pour ne pas avoir ces problèmes. Il s'agira d'une méthode qualitative, qui nécessite moins de précision pour la modélisation. Des approches qualitatives ont déjà été réalisées par le passé en écologie et ont montré leurs atouts et faiblesses, [6] [9], ce que nous allons à présent développer. Cette méthode a déjà été utilisée pour représenter des populations, comme dans Ecomata [10], [18], mais jamais pour modéliser les flux de populations
Dans cette partie, nous allons présenter les caractéristiques des modèles qualitatifs, avec leurs avantages et inconvénients. Ensuite, le formalisme utilisé, celui des automates temporisés, sera développé, avant de nous intéresser au model-checking et la logique TCTL.

2.1 Pourquoi un modèle qualitatif

2.1.1 Présentation des modèles qualitatifs

Un modèle doit s'adapter aux connaissances disponibles sur le système qu'il représente, mais également à la tâche pour laquelle il est conçu (gestion de risques, prévisions, etc..). Un modèle qualitatif se caractérise par la représentation du système en états discrets, appelés *domaines d'abstraction* [14]. Pour une variable continue, ils sont obtenus à partir d'une segmentation du domaine d'existence en un nombre fini d'intervalles (discrétisation de la variable). On définit pour cela des états (ex : population élevée, faible, etc...) et des transitions qui permettent une évolution entre différentes états (ex : passer d'une population faible à moyenne), en définissant les interactions entre les différentes populations (ex : baisse du nombre de prédateurs fait augmenter le nombre de proies). Les domaines d'abstraction peuvent notamment servir à définir l'état de la population, comme c'est le cas pour les espèces animales menacées par l'UICN, Union Internationale pour la Conservation de la Nature. Le véritable

enjeu se situe dans la définition des règles pour représenter ces changements d'états et ce qu'elles impliquent.

2.1.2 Les avantages et les inconvénients

Les avantages : L'un des premiers avantages est de ne pas avoir à déterminer de façon précise (par des équations) les changements d'états et les interactions entre éléments d'un système. La modélisation qualitative fait abstraction d'une partie de la précision du modèle, notamment en lien avec la définition du domaine d'abstraction [15]. Il n'y est pas nécessaire d'expliciter précisément les relations entre populations. Le modèle qualitatif fonctionne donc particulièrement quand les connaissances du système sont incertaines et/ou incomplètes [14]. De plus, de nombreuses contraintes existent suite aux hypothèses réalisées pour utiliser les modèles numériques. Celles-ci ne sont pas toujours vérifiées et il devient alors très compliqué de réaliser une modélisation du système. Il existe cependant des parades, mais elles n intégrent pas les événements rares qui peuvent cependant avoir une influence importante (ex : sinistre naturel comme une tornade). De plus, les modèles qualitatifs n'ayant pas besoin de données précises pour définir l'état initial du système, ceux-ci sont moins sensibles à la qualité des données initiales, qui peuvent influer sur l'évolution du modèle. Si la précision des données recueillies est moins bonne que celle du modèle, cela peut aboutir à des résultats erronés. Une volonté d'avoir une trop grande précision peut aussi mener à une perte de cette précision suite à une mauvaise connaissance des processus et aboutir à des résultats erronés. Avec un modèle numérique, il peut donc être dangereux de rechercher une grande précision, surtout si cela inclut des hypothèses contraignantes sur le modèle, pas toujours respectées.

Un autre avantage est qu'une seule utilisation d'un modèle qualitatif équivaut à une infinité de scénarios en modèle quantitatif [14]. En intégrant l'aléatoire dans un modèle numérique, un très grand nombre de chemins peuvent mener à une même situation finale, alors qu'un modèle qualitatif les intégrera en une seule simulation. Il est donc possible d'économiser du temps et des moyens pour comparer les issues possibles. De plus, un modèle qualitatif fournit en général le chemin d'évolution suivi par le système pour arriver à l'état final, alors que de nombreux calculs intermédiaires sont nécessaires en modèle quantitatifs, ce qui est coûteux ne serait-ce qu'en termes de temps. Les interprétations liées à ces sorties en sont donc plus intuitives. En l'effet l'information est plus facilement assimilable étant donné la nature des sorties (chemin suivi, état à un instant t).
Enfin, il est bien plus aisé pour un public sur le sujet de s'approprier un tel modèle, car son raisonnement est plus proche de celui de l'homme que les formalismes mathématiques. Aucune connaissance approfondie n'est nécessaire pour comprendre le système d'évolution, une fois celui-ci créé (ex : fait de comprendre qu'une hausse du nombre de prédateurs entraîne une baisse de la quantité de proies).

Ce ne sont pas cependant des modèles qui ne présentent que des avantages par rapport aux modèles numériques classiques. Les modèles qualitatifs possèdent aussi des inconvénients.

Les inconvénients : Dans certains modèles où persistent des variables continues [17], les modèles qualitatifs ne sont pas adaptés. Il est difficile de discrétiser certaines variables continues en un nombre fini d'états. Le nombre d'états nécessaire est alors trop important, et le modèle devient irréalisable. Dans les cas où un degré de précision élevé est nécessaire, le modèle qualitatif est également peu adapté. Les domaines d'abstraction ne permettent pas une précision suffisante si on veut éviter un trop grand nombre de domaines d'abstraction.De plus, il peut être difficile de réaliser ces domaines d'abstraction, notamment dans le cas de variables multidimensionnelles (problème pour transition entre états).

Il est également compliqué de réaliser des comparaisons entre modèles qualitatifs de façon générique, notamment en écologie pour ce qui est des différences de vitesse de développement. Or de telles informations peuvent être primordiales pour la prise de décision [7]. C'est le cas quand il faut choisir un modèle prédictif qui servira de base après. Dans notre cas, l'aide à la décision aura pour objectif d'améliorer l'état global d'un écosystème selon plusieurs critères, en menant les actions les plus adaptées à ces objectifs.
Enfin, les représentations graphiques liés aux modèles qualitatifs sont moins simples à lire que celles des modèles numériques. Ceci est lié aux domaines d'abstraction car la continuité entre ces domaines est moins évidente que pour des valeurs numériques. Il existe en plus de nombreuses formes graphiques permettant de représenter des résultats numériques auxquels le public est habitué pour les modèles numériques(histogramme, graphique, tableau, etc...), ce qui accentue encore ce point.

2.1.3 Représentation de processus spatiaux

Il est également possible de représenter des éléments spatiaux, comme un paysage agricole, à l'aide de modèle qualitatif [16], même si peu de références portent sur ce point. Pour cela, on crée un découpage spatial avec les différentes zones, à l'aide d'un ou plusieurs critères. On réalise alors des diagrammes permettant de simplifier la représentation mais qui permettent aussi de connaître touts les liens entre les différents éléments (quels sont les objets qui ont des liens spatiaux). L'enjeu est alors de représenter les liens entre les éléments spatiaux, comme par exemple sur la durée. Dans notre cas, il s'agit de modéliser les migrations d'individus entre les populations de différentes entités spatiales, notamment dans les règles de décision pour les transitions (qui seront évoquées plus tard dans ce document).

2.1.4 Représentation d'un écosystème marin : EcoMata

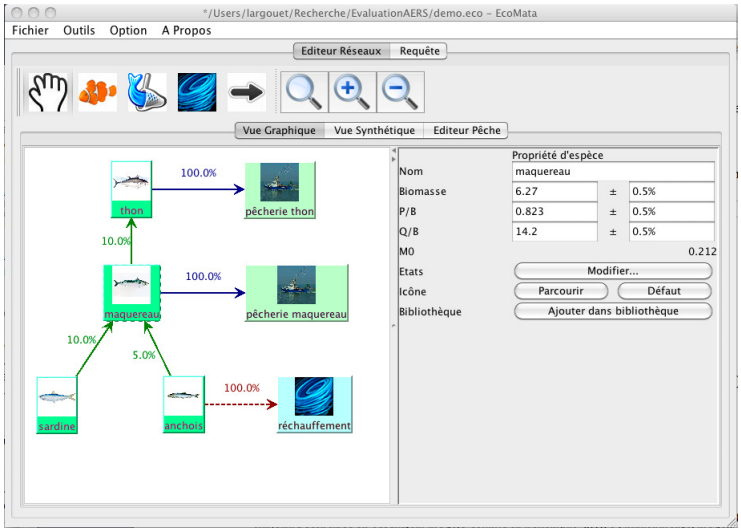


FIGURE 1 – Interface d'EcoMata

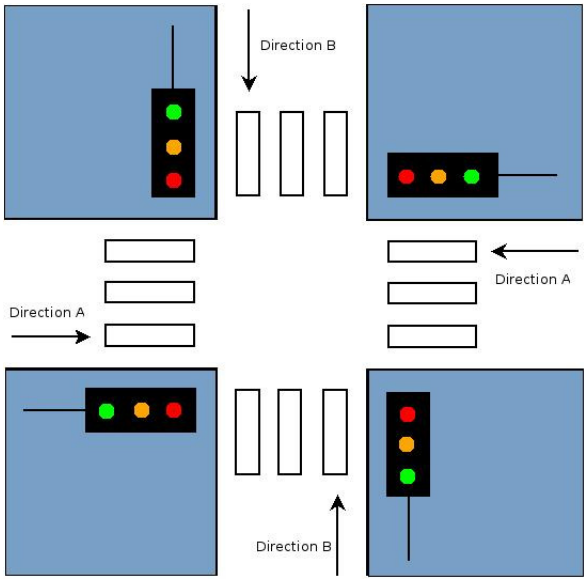
Une première approche, utilisant des méthodes de modélisation qualitative, a été développé comme outil d'aide à la décision sur des domaines d'applications en écologie. EcoMata permet de modéliser de façon qualitative un écosystème marin soumis à une pression de pêche. **EcoMata** a pour objectif de modéliser des liens trophiques entre espèces marines (thon, sardine, etc...) qui sont mal connus dans la littérature, ou difficiles à évaluer [8], [10], [18] pour des populations de tailles importantes, avec de nombreuses interactions. Ce logiciel permet de modéliser les évolutions des populations dans

le temps, en tenant compte des interactions inter-espèces, mais aussi de facteurs environnementaux (réchauffement climatique) et des pressions de pêche exercées sur certaines espèces. Un des objectifs développés est de trouver le plan à mettre en place pour maximiser la force de pêche sur le temps sans pour autant atteindre les stocks de poisson (éviter des états de populations trop médiocres). C'est ce qui est présenté en Figure 1, où un environnement à trois niveaux trophiques est proposé, avec une force de pêche présente sur les deux niveaux les plus élevés.

Pour permettre la représentation des ces populations au sein d'Ecomata, une méthode bien spécifique a été utilisée pour modéliser les évolutions des différentes espèces. Nous allons à présent voir ce que sont des automates temporisés, la méthode utilisée dans Ecomata.

2.2 Les Automates temporisés

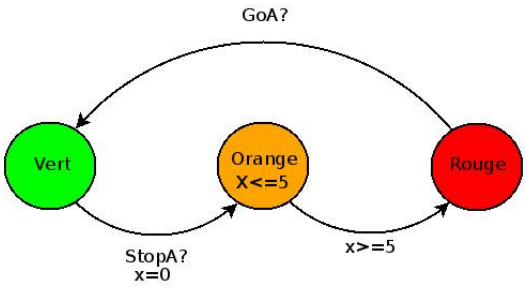
Les automates temporisés ont été proposés par Alur et Dill [2] pour des modèles en temps réel. Il s'agit d'automates, définis à partir d'un nombre d'états finis (issus des domaines d'abstraction), qui sont capables de manipuler des horloges qui évoluent de manière continue et avec un même pas de temps. Une transition entre deux états est limitée par une garde, condition(s) à vérifier pour que la transition puisse avoir lieu (soit temporelle, soit sur l'état du système). Chaque localisation peut posséder un invariant, qui limite les valeurs possibles d'une ou plusieurs variables, notamment les horloges, pour éviter qu'un état ne perdure dans un état (un invariant est une condition à vérifier pour pouvoir rester dans la localisation actuelle). Des définitions plus précises sur le fonctionnement des automates et leurs interactions sont développées en Annexe 1. L'outil utilisé pour réaliser ces automates sera Uppaal, un outil de recherche en développement. Son fonctionnement est présenté en Annexe 2.



Source: P. Gaignon
FIGURE 2 – Organisation simplifié d'un carrefour

Pour illustrer le sujet, nous allons utiliser l'exemple-jouet de la Figure 2. On y modélise un carrefour avec des feux de circulations. L'enjeu est que les voitures ne puissent s'engager que depuis une seule direction à la fois, A ou B, afin d'éviter tout accident. La direction étant ici évoquée d'un point de vue mathématique, elle se décline donc en deux sens. Il faut donc synchroniser les feux pour organiser le flux de véhicules. Dans notre cas, on supposera que les feux situés dans les deux sens de la

direction A seront synchronisés et seront dans le même état à tout instant. On s'intéresse uniquement aux variations selon les deux directions possibles, A et B. On commencera par expliquer le principe de fonctionnement avec de résumer les constituants d'un automate.



Source: P. Gaignon
FIGURE 3 – Automate d'un feu de circulation

Chaque feu est contrôlé par un automate, représenté dans la Figure 3. Celui-ci peut être dans trois localisations, correspondant aux trois couleurs d'un feu classique. Dans l'exemple sera présenté le cas d'un feu situé sur la direction A. On définit un état comme la combinaison de la localisation et des horloges d'un automate. On part d'un état initial où tous les feux sont rouges par sécurité, avec toutes les horloges initialisées à zéro. Quand le feu reçoit le message *GoA* (une synchronisation), il passe alors au vert. Il reçoit alors ensuite un autre message *StopA*, qui le fait passer à la localisation *Oragne*, en mettant son horloge à zéro. Il ne peut rester à cette localisation que pour un temps inférieur ou égal à 5, condition à vérifier avant d'obliger une évolution de l'état (invariant). Il doit également vérifier une condition (garde) pour évoluer (transition) et repasser vers la localisation Rouge.

Nous avons donc présenté comment fonctionne un feu indépendamment de tout autre élément. Cependant, ses variations d'état, c'est-à-dire de couleurs, sont dépendantes de messages reçus et envoyés (synchronisations). Il y a donc un automate chargé de ces modifications pour l'ensemble des feux. Celui-ci est présenté dans la figure 4. C'est ce contrôleur qui organise la circulation globale du carrefour. L'état initial se définit par la localisation *Arrêt*, celui en bas de la figure où tous les feux sont rouges, et toutes les horloges à zéro . Au bout de 45 unités de temps, il envoie le message *GoA* qui fait passer au vert les deux feux situés sur la direction A. Il y a initialisation de son horloge au même instant, elle prend donc zéro comme valeur. Au bout de 40 secondes (temps imposé par la garde et l'invariant), il envoie un message *StopA* qui fera passer à l'orange puis au rouge les deux feux situées sur la direction A. 5 secondes plus tard, il va alors envoyer un message *GoB*, faisant passer au vert les feux sur la direction B. Il s'agit donc d'un cycle répété régulièrement pour ordonner la circulation. Il s'agit bien sûr d'un exemple simplifié, d'autres actions pouvant intervenir dans des carrefours réels.
Un automate se compose donc des éléments suivants :

des localisations : ce sont les modalités qualitatives que peut prendre le système (les niveaux de population dans notre cas par exemple)

des horloges : qui mesurent le temps dans le système et qui évoluent de manière syncrhone

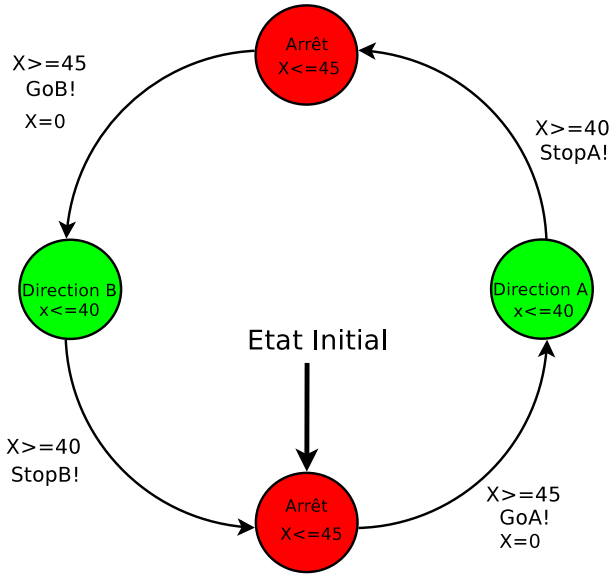
un état : la combinaison de la localisation d'un automate et de ses horloges

des invariants : des conditions à respecter pour rester sur une localisation donnée (Par exemple, une contrainte temporelle)

des transitions : qui permettent à un automate de passer d'un état à un autre, soit par écoulement du temps, soit par un changement de localisation

des gardes : des conditions à vérifier pour réaliser une transition

des synchronisations : des canaux permettant des échanges d'informations entre plusieurs automates. Les différentes types de synchronisations sont présentées en début d'Annexe 6.



Source: P. Gaignon
FIGURE 4 – Automate d'un contrôleur de feux

2.3 Model Checking et TCTL

Model checking : Le model-checking est une technique utilisée pour l'interrogation des automates temporisés. Il permet de tester les propriétés du système (évolution, coexistence de localisations et d'horloges) que peut posséder un modèle établi à l'aide de logique temporelle. A partir d'un modèle initial et des règles de transitions définies, des requêtes sont établies pour tester des caractéristiques possibles du système, notamment exprimées en logique temporelle.
L'objectif est de tester les évolutions possibles d'un système élaboré graphiquement pour ses états et transitions (voir Annexe 2 à propos d'Uppaal). On écrit des requêtes en interrogeant le système sur ses évolutions possibles.
Pour tester les requêtes établies, une des formes de logique les plus utilisée est Computation Tree Logic (CTL), avec sa forme intégrant les requêtes sur le temps, TCTL (pour Timed CTL).

Timed Computation Tree Logic : TCTL est une logique utilisée pour écrire les requêtes adressées par model-checking, où le temps est considéré comme une variable continue et non plus discrète. Il s'agit d'une forme de logique incluant notamment les éléments temporels [5], [11]. Elle est issue de la logique CTL. On décrit donc le système à l'aide de proposition atomiques (les localisations du systèmes qui nous intéressent), des opérateurs booléens ($\wedge, \vee, \longrightarrow, \neg$) et des modalités ($E, A, G, F, X$).
Les modalités existantes sont :

- E ou $\exists$: il existe
- A ou $\forall$: Pour tous, quelque soit
- G ou $\Box$: Globalement (toujours vérifié)
- F ou $\diamondsuit$: Finalement (on finit toujours par atteindre)
- X ou $\bigcirc$: Ensuite (il en découle aussitôt après)

Elle est complétée en ajoutant à l'opérateur temporel U (pour until) des contraintes de la forme "$\bowtie c$", avec $\bowtie \in \{=, <, >, \leqslant, \geqslant\}$ et $c \in \mathbb{N}$. Il est possible de composer avec les modalités précédentes, donnant par exemple $E_U_{\bowtie c}_$.

Par exemple, on veut tester la déclaration suivante : "*Une population d'insectes peut perdurer à un état élevé pendant 12 jours au minimum avant d'utiliser des pesticides*", qui va se traduire par :

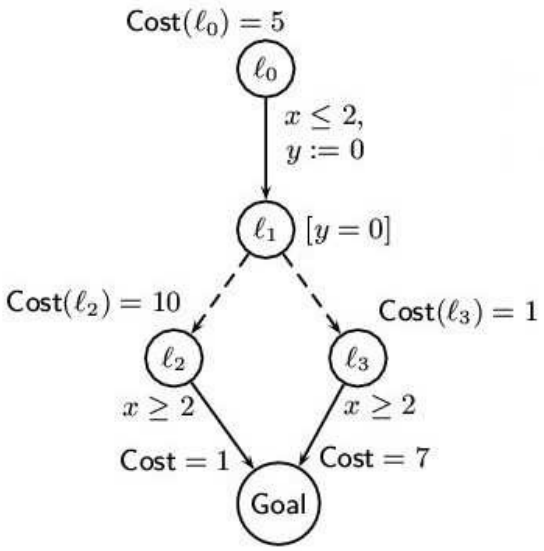
$$\mathrm{E}_{Pest_{high}} \mathrm{U}_{x \geqslant 12} pesticide$$

Suite à cela, il va y avoir une utilisation de pesticide qui va affecter la population selon la propriété suivante : "*La population d'insectes passe à un état faible moins de 5 jours après l'utilisation de pesticide*". Elle va alors s'écrire sous la forme :

$$\forall\Box(pesticide \longrightarrow \forall\Diamond_{x \leq 5}(Pest_{low}))$$

Elle permet notamment d'exprimer les états du système que l'on souhaite tester, selon que la condition est toujours vérifiée quelque soit le chemin à un instant donné ou à tout moment. On peut par exemple tester si des états peuvent cohabiter, si un état peut en induire un autre de façon systématique. On obtient dans tous les cas comme réponse un booléen, la requête TCTL étant vérifiée ou non.

2.4 Notion de coût en model-checking



Source: Patricia Bouyer, Franck Cassez, Emmanuel Fleury, Kim G. Larsen

FIGURE 5 – Exemple d'automate temporisé avec coût

Une notion qui peut être importante est la prise en compte d'un coût dans l'évolution d'un système [4]. Des chemins peuvent mener à un même état du système en utilisant des évolutions différentes. En effet, ces évolutions du système peuvent être de nature diverse, et avoir des coûts plus ou moins importants et de natures différentes (économiques, temporelles, environnementales, etc..). Le chemin devient alors important lors de l'atteinte d'un état du système.
C'est ce qui est développé dans la Figure 5. Si l'on a pour but d'atteindre l'état *Goal* en partant de l_0, deux chemins s'offrent à nous. Si l'on emprunte celui passant par l_2, on obtient un coût total de 11, alors qu'en passant par l'état l_3, le coût final est seulement de 8. Dans les deux cas, on obtiendra la même situation finale, mais le coût n'aura pas été le même selon le chemin emprunté.

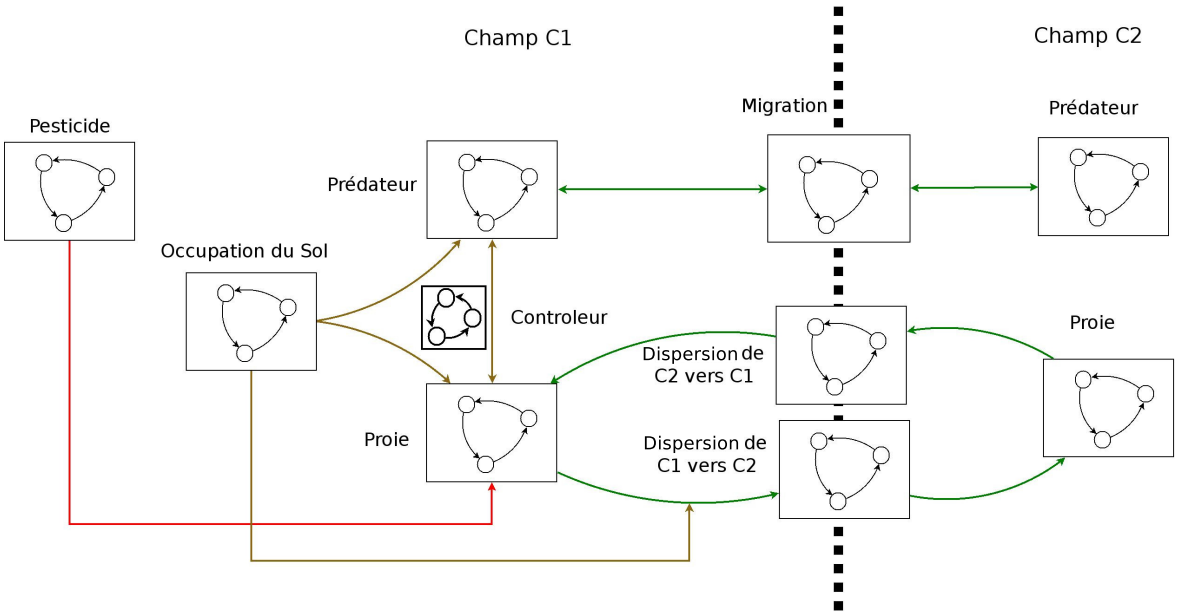
Après avoir présenté le fonctionnement et les particularités de ces automates, nous allons présenter ceux qui ont été utilisés pour le modèle.

3 Création des automates constitutifs du modèle

Comme évoqué précédemment, nous cherchons les évolutions des populations au sein d'un paysage agricole en évolution. Pour cela, il est donc nécessaire de représenter ces populations, proies et prédateurs, pour modéliser leur évolution. De plus, nous intéressant à des enjeux agricoles, il faut aussi représenter la possibilité d'utiliser des herbicides pour représenter le rôle des agriculteurs. Toutefois, le paysage agricole doit pouvoir évoluer, afin de pouvoir tester différentes évolutions. Pour cela, il est nécessaire de séparer le paysage en plusieurs éléments (ou constituants). Pour chaque élément, il faudra donc représenter l'occupation du sol, mais aussi les populations qui y sont présentes. Enfin, les populations n'étant cantonnées dans un seul élément du paysage, il est nécessaire de créer des automates permettant le déplacement des individus au sein du paysage. Les prédateurs se déplacent selon un processus de migration entre éléments contigus, alors que les proies se propagent par dispersion sur des éléments qui peuvent être plus distants.

En conclusion, nous allons créer des automates temporisées pour représenter les éléments suivants :

— Des automates de population, proie et prédateur, pour chaque élément de paysage
— Des automates pour l'occupation du sol et pour les pesticides pour chaque élément de paysage
— Un automate contrôleur par élément de paysage pour y faciliter la communication
— Des automates entre éléments contigus pour représenter la migration des prédateurs
— Des automates pour modéliser la dispersion des proies dans tous le paysage



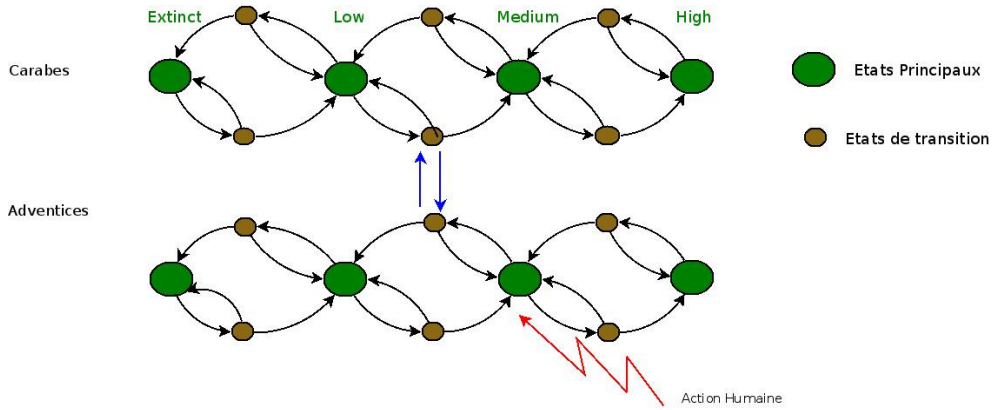
Source: P. Gaignon
FIGURE 6 – Modèle global désiré

Ces éléments sont résumés en Figure 6, où les liens entre les différents liens entre automates sont également représentés. Ces liens montrent les axes de communication majeures au sein du système. Cela est rendu possible par l'utilisation de synchronisation, ou de variables globales du système (non spécifique à un automate). Par exemple, il existe un vecteur contenant l'état de toutes les populations de proies à l'aide d'un encodage numérique. Il est donc possible à chaque automate d'interroger ce vecteur pour connaître l'état d'une population sur un élément donné. La Figure 6 présente un exemple

simplifié avec seulement deux éléments de paysage, et qui ne montre pas la possibilité d'une dispersion sur de plus grandes échelles.

3.1 Les Automates de Populations

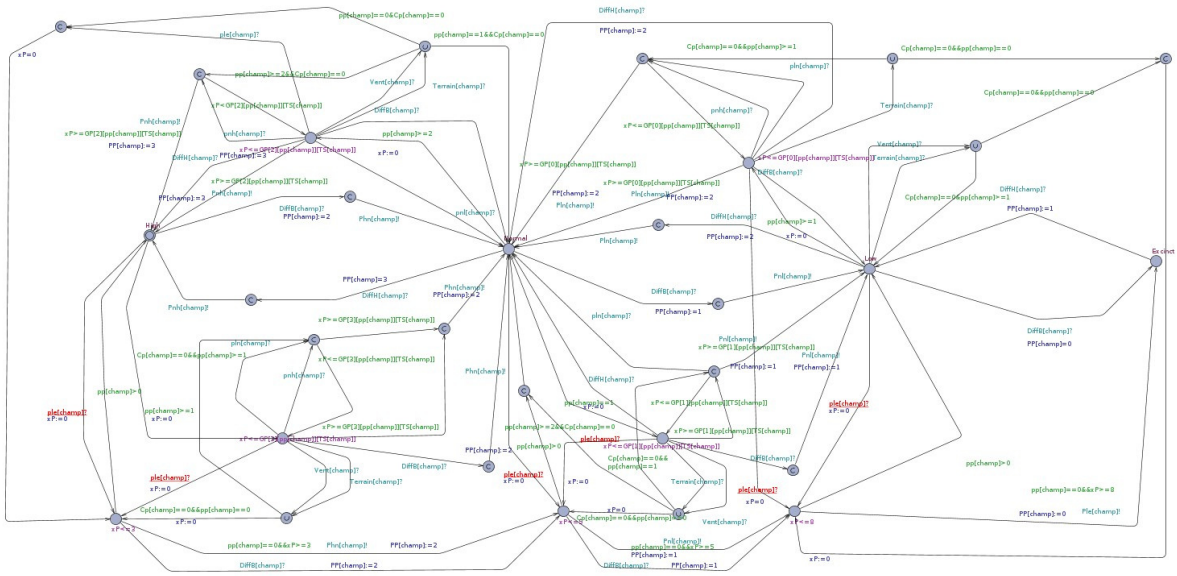
Les bases dans les règles d'évolutions sont les équations de Lotka-Volterra, présentées dans les équations 1 et 2. Les automates représentent les fluctuations de populations situées dans chaque élément de paysage, et permettent les liens entre les proies et les prédateurs. La figure 7 présente les principaux états des deux automates. Les populations sont caractérisées par quatre états "principaux", correspondant aux états de populations, à savoir *High*, *Medium*, *Normal* et *Extinct*. Les autres localisations de l'automate sont des états qualifiés de transitoires, car il n'est pas possible pour le système d'y rester indéfiniment. En effet, soit il existe des contraintes temporelles obligeant à les quitter, soit par leurs caractéristiques qui obligent qu'elles soient quittées avant toute autre évolution du système. Pour cela, on utilise des invariants pour empêcher que l'automate persiste dans un état de transition, mais aussi des états appelés *urgent* et *commited* qui arrêtent le temps du système et empêchent à d'autres éléments d'évoluer tant que ces états n'ont pas été quittés. Pour rappel, ces éléments sont définis en Annexe 2, qui concerne l'utilisation Uppaal.



Source: P. Gaignon

FIGURE 7 – Représentation des évolutions de population et interactions entre espèces

Il est évident que, comme pour Lotka-Volterra, les évolutions des espèces sont interdépendantes. Par exemple, une hausse du nombre de proie facilitera des hausses de la population de prédateur. De plus, les proies sont soumises à l'action humaine dans les champs, sous forme de pesticide si leur population devient trop importante. La communication entre les automates est donc très importante, que ce soit par l'utilisation de messages directs, des synchronisations, ou par la mise à jour de variables globales dans le système. L'exemple de l'automate du prédateur créé sous Uppaal est présenté en figure 8, afin d'en montrer la complexité. Il est possible d'y distinguer les 4 états principaux, ainsi qu'une multitude d'états de transitions, pour répondre à toutes les évolutions du système. Les 4 états principaux correspondent aux états de population décrits précédemment (*High*, *Medium*, *Normal* et *Extinct*). Les temps de transition entre ces états principaux dépendent en plus de l'occupation du sol et de l'autre espèce présente. Ces temps sont obtenus par l'interrogation des paramètres définis lors du paramétrage du modèle. Ce paramétrae a été permis à l'aide d'un modèle simplifié de type Lotka-Volterra. Juste en modifiant ces paramètres d'entrée du modèle, il est donc possible de faire évoluer le comportement de l'automate sans avoir à le retravailler intégralement. L'ensemble des paramètres utilisés est défini en Annexe 3.
Chaque automate représente donc une population, de proies ou de prédateurs et gére ses évolutions.



Source: P. Gaignon

FIGURE 8 – Automate de Population du Prédateur, exemple de la complexité des automates

Cela se fera notamment en communiquant avec les autres automates du système selon les axes d'évolutions (lien de prédation, déplacements d'individus, action humaine).

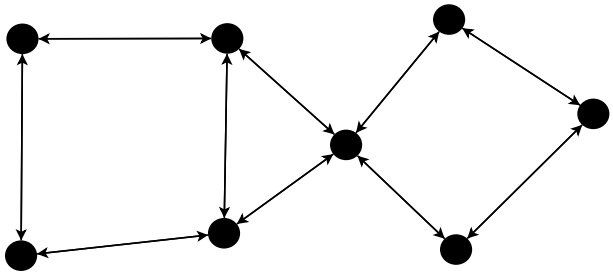
3.2 Les Automates de Paysage

Comme évoqué précédemment, nous nous intéressons au rôle du paysage agricole sur les populations, mais également à son hétérogénéité. Il faut donc représenter le paysage, en définissant des éléments d'un point de vue qualitatif. Pour cela, nous recherchons des limites pour définir des zones globalement homogènes du point de vue de l'occupation du sol. Quatre occupations du sol ont ainsi été dégagées :

- les champs cultivés
- les champs en période inter-culturale (PIC)
- les habitats semi-naturels (HSN),
- les matrices (route, zone habitée, etc..)

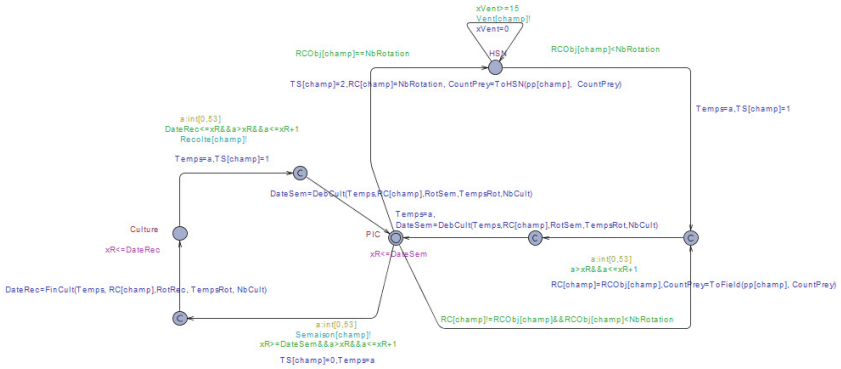
Pour des questions de simplification, le paysage étudié sera de type lattice, c'est-à-dire une grille de forme carré. Il faut à présent définir les positions des différents éléments dans le paysage. Dans ce cadre, nous définissons des relations de proximité. Nous cherchons donc les éléments contigus dans le lattice. Cette proximité est illustrée en figure 9, où les flèches indiquent les éléments contigus. La distance entre deux éléments se définit donc par le nombre de flèches minimal à emprunter pour aller d'un élément à un autre. Dans un lattice carré, cela correspond donc au nombre minimal de déplacement horizontal et vertical pour relier deux points, c'est la distance de *Manhattan*. Sur la figure 9, le point central se trouve alors à deux unités de distances des points qui lui sont le plus éloignés. Les deux points les plus éloignés se trouvent quant à eux à quatre unités de distance.

Comme évoqué précédemment, les occupations du sol évoluent dans le temps, ce qu'il faut donc représenter. Pour des raisons de simplification, il a été décidé que les éléments de type matrice seraient inamovibles dans le temps, en raison d'un temps d'étude assez court (inférieur à cinq ans). Il n'est



Source: P. Gaignon
FIGURE 9 – Représentation des éléments paysagers

donc pas possible qu'un élément devienne une matrice ou qu'une matrice voit son occupation du sol évoluer. Pour cela, les matrices ne seront pas représentées dans les automates, car elles sont invariantes. Ce seront des variables globales du système qui suffiront à porter l'information.



Source: P. Gaignon
FIGURE 10 – Automate d'occupation du sol d'un élément du lattice

L'automate représente les évolutions d'occupation du sol est présenté en figure 10. Nous pouvons y distinguer les trois évolutions possibles, à savoir la culture, le sol nu, correspondant à la période inter-culturale (PIC) et l'HSN. Nous distinguons du coup deux éléments, les HSN et les champs qui alternent entre deux états, culture et PIC. Il n'est possible pour un automate d'évoluer, hors l'alternance liée aux cultures, que si cela lui permet d'atteindre la dynamique paysagère que l'on cherche à tester. Cette contrainte permet de limiter le nombre d'évolution du système, mais aussi de se rapprocher à la réalité où l'occupation ne peut être modifiée trop fréquemment. En ce qui concerne l'évolution liée aux cultures, il est possible de paramétrer le modèle pour intégrer des rotations culturales différentes. Chaque rotation correspond à un objectif distinct, car elles peuvent avoir des impacts différents sur les espèces présentes. Il existe dans ce but des fonctions chargées d'assurer les rotations culturales. Elles doivent retrouver les gardes et les invariants pour chaque culture, pour représenter les alternances entre semis et récoltes dans chaque élément paysager, à la suite de leur alternance, selon un schéma donné. Ces fonctions, écrites en langage C, sont présentées en Annexe 4. Pour des soucis de simplification, notre modèle ne contiendra qu'une seule rotation à une seule culture pour simplifier le modèle et les temps de simulation.
Pour chaque élément de paysage, il existe un automate dit *contrôleur*. Il a pour rôle de faciliter les communications entre les différents automates présents dans chaque élément géographique du lattice. Il facilite l'interprétation des synchronisations utilisées, en résumant certaines informations. Par

exemple, les synchronisations *Récolte* et *Semaison* amènent toutes les deux des modifications de l'état du sol, ce qui est résumé par la synchronisation *Terrain*. Son rôle peut être aperçu en figure 11, dans les communications au sein du système.
Pour chaque élément paysager de type non matriciel, il existe un autre automate pour réguler l'utilisation des pesticides dans le paysage. Il peut donc déclencher leur utilisation si les proies, les nuisibles, sont présentes en quantité correspondant à un état *High* sur une culture. Si les conditions restent vérifiées pendant un temps défini, cela déclenche l'utilisation des pesticides qui affecte directement la population des proies.

3.3 Les Automates de Migration et Diffusion

L'un des enjeux de ce travail est la représentation des déplacements d'individus au sein de paysage, que ce soit contrôlé, la migration des carabes, ou non, la diffusion des adventices.
Dans le cas des prédateurs, la migration se réalise entre deux éléments contigus du paysage. De plus, elle ne peut se réaliser que si certaines conditions sont réunies, et son déclenchement reste aléatoire dans ce cas là. Une migration peut donc réunir les conditions nécessaires sans pour autant se réaliser. Cette possibilité dépend de l'état de populations des deux éléments géographiques contigus, et de leur occupation du sol respective. De plus, si plusieurs migrations sont possibles incluant un même élément paysager, il existe un ordre de priorité dans l'exécution des migrations. Ces règles de priorité s'appliquent quand plusieurs migrations se déclenchent pour un même élément. L'ensemble des règles sont présentées en Annexe 5. L'ensemble des règles de migration est résumée dans le tableau 1. Dans ce tableau, une migration est possible si la valeur est différente de 0, et plus elle est éloignée de zéro, plus la migration est prioritaire. Le signe indique si la migration s'effectue depuis N1 vers N2. Un signe indique de N1 vers N2, et inversement. Les valeurs présentés dans le tableau 1 ne sont que des valeurs relatives, permettant de situer le degré de priorité entre plusieurs migrations.

TABLEAU 1 – Priorités de Migration chez les Prédateurs

	N1	Culture				Inter-Cult/HSN				Matrice			
N2		H	N	L	E	H	N	L	E	H	N	L	E
Cult	H	0	0	-8	-9	0	0	-3	-7	0	0	0	-2
	N	0	0	0	-7	3	0	0	-4	0	0	0	0
	L	8	0	0	0	5	4	0	0	0	0	7	0
	E	9	7	0	0	7	6	5	0	0	0	8	0
PIC / HSN	H	0	-3	-5	-7	0	0	-4	-8	0	0	0	-3
	N	0	0	-4	-6	0	0	0	-6	0	0	0	0
	L	3	0	0	-5	4	0	0	0	0	0	6	0
	E	7	5	0	0	8	6	0	0	0	0	7	0
Matrice	H	0	0	0	0	0	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0
	L	0	0	-7	-8	0	0	-6	-7	0	0	0	-1
	E	2	0	0	0	3	0	0	0	0	0	1	0

Le déplacement des proies, la dispersion, ne peut être modélisé dans notre cas de la même façon, car étant des graines, elles ne peuvent se déplacer seules. Leurs mouvement sont dépendants d'événements extérieurs. Il existe trois événements possibles, à savoir au moment des récoltes et des semis pour les champs, et une dispersion par le vent pour les HSN. Ces facteurs de dispersion permettent en plus des déplacements d'individus vers des éléments géographiques non contigus. Il existe un automate pour chaque sens de dispersion de graine. Par exemple, un champ qui peut émettre vers N autres éléments,

engendrera 2*N automates de dispersion, N pour lesquels il est émetteur, et N pour lesquels il est récepteur. Suite au déclenchement d'un événement pouvant mener à une dispersion, le système tire une valeur aléatoire entière dans un intervalle donné, entre 1 et 20 dans notre cas. La dispersion peut avoir lieu si la valeur tirée est strictement inférieure à la probabilité que la dispersion ait lieu, ramenée sur notre intervalle. Une probabilité nulle équivaut donc à une dispersion impossible. La probabilité dépend des états de populations de l'émetteur et du récepteur, mais aussi de la distance. En effet, plus la distance est grande, plus la probabilité sera faible pour un même cas de populations. Dans certain cas, les dispersions peuvent en plus mener à des variations de plus d'un niveau de population.
Lors de la réalisation de ces dispersions, le temps est arrêté dans le système, empêchant toute évolutions des autres automates, comme par exemple pour les autres proies et les prédateurs. Le temps reprend lorsque tous les cas de dispersion ont été résolus, et que les états de populations des proies ont été mis à jour.

3.4 Les variables globales et le paramétrage du système

En conclusion, on peut représenter le schéma de communication entre tous les automates, comme sur la figure 11, avec toutes canaux de communications existants. Une liaison verte indique un *broadcast chan*, une communication avec un émetteur et un nombre non définis de récepteur (de zéro à infini selon les cas). Par exemple, la synchronisation *Terrain* indique une modification de l'occupation du sol, ce qui peut affecter les automates de proies et de prédateurs, si des transitions sont en cours, mais aussi les automates de migrations si des flux de populations sont en cours. Une flèche rouge indique un *chan*, message avec un émetteur et obligatoirement un seul et unique récepteur (L'émetteur ne peut évoluer et effectuer cette évolution s'il n'y a pas de récepteur). Par exemple, la synchronisation *Pest* indique qu'il vient d'y avoir l'utilisation de pesticide sur les proies, et que leur niveau de population doit forcément diminuer. Le rôle de chaque message est défini en Annexe 6. Il faut en plus ajouter l'utilisation de l'ensemble des variables globales utilisées par le système pour faciliter des communications indirectes, comme par exemple, les niveaux de population ou les occupations du sol.

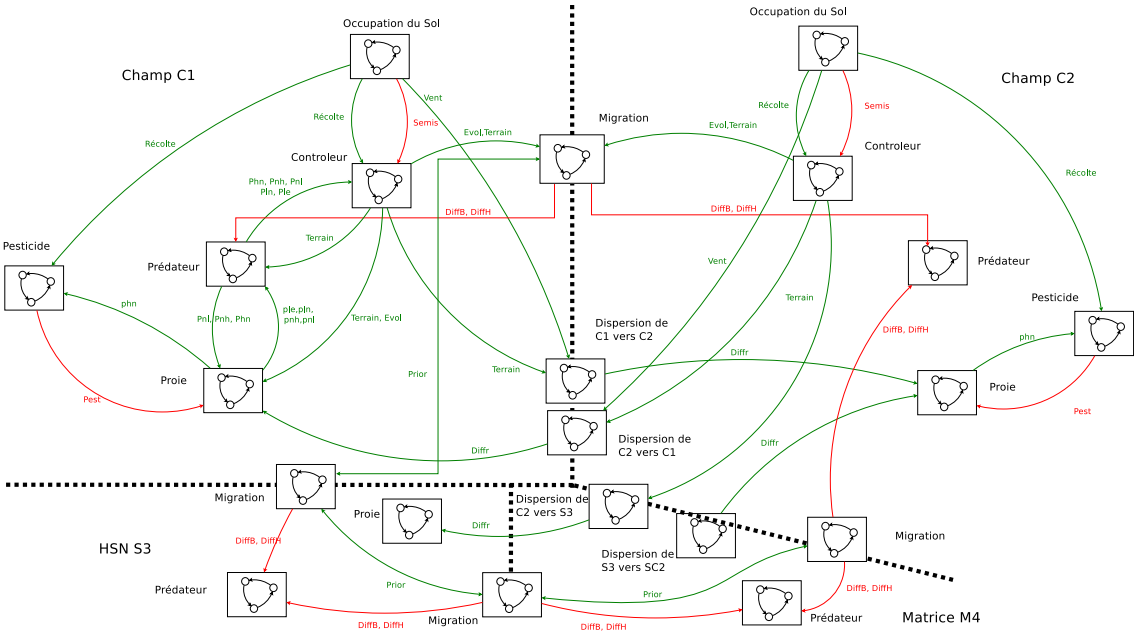


FIGURE 11 – Représentation des communications entre les différents automates du modèle

Ces variables globales facilitent les interrogations de l'état d'un automate. Par exemple, quand on veut connaître l'état d'une population qui se décline en quatre état, la variable porteuse de cette information limite le nombre de possibilité par rapport à l'automate. En effet l'automate comporte de nombreux états de transitions, mais qui correspondent à de même états de populations, mais augmenteraient le nombre de cas possible s'ils étaient utilisés à la place des variables (une vingtaine contre les quatre modalités de la variable.

4 Contraintes liées aux automates

4.1 Problème de taille de paysage

Un des enjeux est la modélisation d'un écosystème de taille importante. Comme pour tout écosystème, plus l'élément modélisé est grand, plus sa complexité est importante. On peut essayer d'approximer le nombre d'automates présent dans un lattice de côté L, soit avec L^2 éléments de paysage. Pour une simplification de l'estimation, on considère que tous les automates sont présents, c'est-à-dire qu'on suppose qu'il n'y a pas de matrice dans le paysage. Cet élément est présentée dans l'équation 3. A noter que cette formule est valide à partir d'un lattice d'au moins de côté 3, en raison d'une distance de dispersion des proies de 2 au sein du modèle.

$$\begin{aligned} Nb_{automates} &= L^2 \textit{ automates de Predateurs } + L^2 \textit{ automates de Proies} \\ &\quad + L^2 \textit{ automates d'occupation du sol } + L^2 \textit{ automates de Pesticide} \\ &\quad + L^2 \textit{ automates controleur } + 2L(L-1) \textit{ automates de migration} \\ &\quad + [2L(L-1) + 2L(L-2) + 4(L-1)^2] \textit{ automates de dispersion} \\ &= 5L^2 + 4L(L-1) + 4L(L-2) + 4(L-1)^2 \\ &= 17L^2 - 20L + 4 \end{aligned} \tag{3}$$

A partir de l'équation 3, on obtient pour un lattice de taille 5*5, 329 automates, pour un lattice 7*7, 697 automates et un 10*10, 1504 automates au sein du modèle. La hausse de la complexité du modèle est donc exponentielle. Pour éviter un problème de taille, on représentera un lattice de taille 5*5 pour limiter la complexité du modèle, avec au total 329 automates. Cette limitation de la taille du lattice permet de conserver des phénomènes de dispersion sur une distance de 2 vers plusieurs automates pour représenter le phénomène de dispersion. Pour rappel, la figure 11 présente l'ensemble des communications et des automates présents dans le modèle. Pour ces mêmes raisons, on limitera la distance des dispersions possibles à 2, pour limiter une explosion du nombres d'automates, même si le modèle peut accepter des distances bien plus importantes. Le paysage représenté serait donc de quelques hectares, ce qui est assez faible à l'échelle d'un paysage agricole, mais très important pour les populations représentées.

4.2 Limitation du nombre de transition

Uppaal ne gère pas les automates en ayant un écoulement continu du temps, ce qui serait trop compliqué. Au lieu de cela, il situe les horloges dans des intervalles de temps, qu'il fait évoluer suite aux transitions effectuées. Il établit de plus des relations entre les différentes horloges pour assurer qu'elles restent synchrones. Il peut donc à partir d'un état envisager l'ensemble des transitions possibles, avec ou sans écoulement de temps préalablement nécessaire pour les réaliser. Cependant, quand on tire une variable aléatoire, comme c'est le cas pour la dispersion des proies, il crée une transition par valeur possible aléatoire.

Si l'on prend un cas d'émission avec un intervalle entre 1 et 100 pour déterminer si la dispersion est possible, la variable issue de ce tirage peut prendre 100 valeurs (Uppaal ne travaille qu'avec des entiers), il en résulte autant de transitions possibles pour une seule dispersion, même si seules deux issues sont possibles. Uppaal considérera chaque transition comme différente du fait de la valeur différente pour la valeur aléatoire. Le nombre de transition reste cependant faible. Toutefois, si on considère le nombre de dispersions réalisées au même temps, N par exemple, alors le nombre de transition possible menant à des états différents est 100^N. Cela augmente grandement l'espace à explorer pour Uppaal, ralentissant son travail. Le nombre de transitions va donc évoluer très vite, et le temps de calcul suit. Il faut donc chercher à diminuer la largeur de l'intervalle au maximum. Pour cela, il est nécessaire de réduire l'intervalle, quitte à perdre un peu de précision mais gagner en performance. Dans notre cas, l'intervalle a été réduit entre 1 et 20, ce qui diminue par 5^N la taille de l'environnement à explorer.

5 Sélection et Analyse d'un échantillon de paysage

5.1 Sélection de Paysages

L'objectif est de tester un large panel d'évolutions de notre paysage initial, pour étudier les facteurs paysages qui influeraient sur les critères établis précédemment. Cependant, il n'est pas possible de tester l'ensemble des dynamiques paysagères, c'est-à-dire toutes les évolutions possibles de notre paysage initial. En effet, si l'on considère notre paysage avec 25 éléments, dont 3 de types matrices, qui ne peuvent évoluer dans le temps, le nombre possible total de dynamiques paysagères est 2^{22}, soit plus de quatre millions au total, pour seulement 22 éléments qui peuvent évoluer. Il faut donc sélectionner parmi l'ensemble des dynamiques paysagères un échantillon représentatif.
Pour cela, on génère d'un point de vue descriptif l'ensemble des dynamiques possibles. Cependant, toutes ne sont pas conservées, les dynamiques devant respecter certaines contraintes. Une dynamique paysagère ne présentant pas ou peu de cultures ne sera pas intéressant car il est nécessaire de conserver une certaine surface agricole utile. De même, il n'est pas possible de modifier toutes les occupations du sol dans le délai d'évolution qui nous intéresse. L'évolution est donc limité en nombre de modifications qui peuvent être apportées au paysage initial. Pour cela, parmi toute les dynamiques existantes, nous ne conserverons que certaines respectant une proportion de cultures située dans un intervalle donné et qui ne sont pas trop éloignées du paysage initial (nombre de modifications limitées).
A partir d'un ensemble de critères paysagers décrits par Leitao [12], spécifiques aux paysages de type lattice, présentés en Annexe 7, nous allons caractériser les dynamiques respectant les critères et rechercher l'échantillon de taille N le plus représentatif de l'ensemble des dynamiques paysagères sélectionnées. Pour cela, nous allons réaliser une ACP sur l'ensemble des dynamiques décrites par ces critères, ce qui permettra d'obtenir la meilleure projection sur l'ensemble de la variété des critères, et d'obtenir des dimensions toutes orthogonales, évidant ainsi des redondances entre certains critères.
Suite à cela, une classification est réalisée sur les résultats de l'ACP. Travaillant sur un exemple restant de taille modeste, nous allons conserver l'ensemble des axes de l'ACP pour réaliser la classification, même si pour des paysages de tailles plus importantes, il pourrait être intéressant de n'en garder qu'une partie pour gagner du temps en termes de calcul. L'objectif est d'obtenir N groupes, correspondant à la taille d'échantillons que l'on peut tester. Il en résulte donc N groupes avec des distances intra-groupes minimales, et des distances inter-groupes les plus élevées possibles. Pour chacun de ces groupes, le premier parangon est extrait, en tant que meilleur représentant de chaque groupe. Nous avons donc N individus, issus des N groupes, qui représentent au mieux la diversité de l'ensemble des dynamiques.
Après avoir sélectionné les dynamiques les plus représentatives, il faut à présent les générer, afin de les tester.

5.2 Génération des Paysages sélectionnés

Il faut donc générer chaque paysage qui sera testé, c'est-à-dire créer un fichier par paysage au format xml, format utilisé par Uppaal. Le paysage initial et les automates étant les mêmes pour chaque paysage, nous allons conserver ces éléments communs et générer tous les paysages de façon automatisés. Pour cela, nous récupérons les parangons obtenus précédemment, les individus les plus représentatifs de chaque groupe, avec leur caractéristiques, c'est-à-dire leur organisation paysagère. A l'aide du logiciel **R** et du package *XML*, il est possible d'extraire les noeuds d'un fichier contenant déjà l'ensemble des automates nécessaires, et de les réutiliser pour d'autres fichiers xml. Un fichier .xml possède une organisation qui s'approche de la ramification d'un arbre. Il existe un premier élément, qui s'apparente au tronc de l'arbre. Ce premier élément, appellé noeud peut donner naissance à plusieurs noeuds, correspondant aux premières branches de l'arbre, qui peuvent à leur tour se ramifier. D'une façon générale, les noeuds correspondent au point de ramification du document. On parle de relation parent-enfant, un noeud parent d

A partir d'un paysage initial, qui sera le même dans toutes les simulations, nous allons tester l'influence des différentes dynamiques paysagères. Pour cela, chaque paysage est décrit comme un objectif à atteindre en termes de simulation, à l'aide d'un vecteur de même taille que celui décrivant les occupations du sol au sein du paysage.

Pour cela, il faut s'intéresser à l'architecture globale d'un fichier créé sous Uppaal. Ceux-ci se composent de la façon suivante, avec quatre noeuds majeurs :

- System : déclaration des variables globales du système.
- Templates : modèles d'automates créés sous Uppaal.
- Declaration : permet de définir l'ensemble des automates qui seront inclus dans le modèle, à partir des modèles créés précédemment.
- Requests : ensemble des requêtes enregistrés à adresser le modèle.

Pour le premier noeud, les variables globales sont les mêmes pour tous les paysages, à l'exception du vecteur chargé d'indiquer la dynamique paysagère au modèle. La partie *Templates* comprend tous les modèles d'automates, qui peuvent être utilisés plusieurs fois avec des paramètres d'entrées différents. Ils sont donc les mêmes pour tous les paysages. Pour ne pas devoir réécrire leur description à chaque noeud, ce qui est très long, on extrait ces automates d'un fichier où ils ont été modelés à l'aide d'Uppaal. Ensuite, la partie *Declaration* ne change pas d'un paysage à l'autre, car il dépend surtout de l'état initial du paysage, qui est le même pour toutes les simulations. Enfin, la partie *Requests* n'est pas ajoutée, car pour gagner du temps, elle seront adressées à partir d'un exécutable et non pas dans Uppaal même pour gagner en performance.

Nous allons à présent nous intéresser aux requêtes qui seront utilisées pour tester le modèle créé et les dynamiques paysagères.

5.3 Requêtes adressées aux modèles

Pour chaque dynamique paysagère, il faut ensuite tester les évolutions qui peuvent se produire et leurs influences. Il existe deux types d'évaluations de ces influences : le coût en pesticides, c'est-à-dire le nombre d'utilisations de pesticides ; et la quantité de nuisibles présents sur les cultures. Pour chaque critère d'évaluation, on sélectionne plusieurs valeurs qui seront testées, correspondant à des importantes relatives, comme par exemple des usages de pesticides importants ou faibles. Pour cela, deux types de requêtes vont êtres étudiées.

Le premier correspond à la possibilité d'atteindre l'objectif de la dynamique en un temps donné, avec des coûts en pesticides limités et une restriction sur la quantité de nuisibles présents sur les cultures. Cette première se décline en trois formes, les deux premières avec un critère d'évaluation et

les objectifs d'organisation du paysage et de temps. La troisième forme, l'équation 6, correspond au fait d'atteindre les deux critères en même temps. Les requêtes obtenues sont donc de la forme suivante (sous forme TCTL) :

$$\exists\Diamond RC = RC_{Obj} \wedge horloge = 250 \wedge CoutPest \leq Crit_{Pesticide} \tag{4}$$

$$\exists\Diamond RC = RC_{Obj} \wedge horloge = 250 \wedge CountPrey \leq Crit_{Nuisible} \tag{5}$$

$$\exists\Diamond RC = RC_{Obj} \wedge horloge = 250 \wedge CoutPest \leq Crit_{Pesticide} \wedge CountPrey \leq Crit_{Nuisible} \tag{6}$$

avec

- RC : l'état du paysage (occupation du sol pour chacun des éléments)
- RC_{Obj} : l'organisation du paysage que l'on cherche à atteindre
- horloge : le temps écoulé depuis le début de la simulation (en semaines)
- CoutPest : Le nombre d'utilisations de pesticides réalisées sur les cultures depuis le début de la simulation
- $Crit_{Pesticide}$: l'objectif qu'on cherche à atteindre en termes d'utilisations de pesticides
- CountPrey : la quantité de nuisibles présentes sur les cultures
- $Crit_{Nuisible}$: la quantité maximale de nuisibles qu'on accepte sur les cultures

Le problème avec ces requêtes est qu'elles ne permettent de savoir que si les objectifs sont atteignables, sans pour autant savoir s'ils sont faciles à atteindre ou pas. Pour cela, le deuxième type de requête cherche à savoir si c'est l'organisation du paysage qui peut impliquer ces résultats. Cela revient à dire que si l'organisation et le temps sont respectés, alors les critères d'évolutions satisferont également les valeurs testées, pour tout chemin de la dynamique paysagère. On reprend l'organisation précédente avec trois formes pour ce type de requête, ce qui donne :

$$\forall\Box(RC = RC_{Obj} \wedge horloge = 250)\ imply\ (CoutPest \leq Crit_{Pesticide}) \tag{7}$$

$$\forall\Box(RC = RC_{Obj} \wedge horloge = 250)\ imply\ (CountPrey \leq Crit_{Nuisible}) \tag{8}$$

$$\forall\Box(RC = RC_{Obj} \wedge horloge = 250)\ imply\ (CountPrey \leq Crit_{Nuisible} \wedge CoutPest \leq Crit_{Pesticide}) \tag{9}$$

Il en résulte un grand nombre de requêtes à poser. Si on veut tester N_1 valeurs pour les pesticides, et N_2 pour les nuisibles, il y a en tout $2(N_1+1)(N_2+1)$ requêtes à tester. Dans notre cas, nous avons pris trois valeurs pour chacun des critères d'évaluation, soit 32 requêtes à tester par paysage. Il n'est donc pas possible de tester les requêtes une par une. Il devient nécessaire d'automatiser leur vérification.

5.4 Automatisation pour la vérification des requêtes

N'étant pas possible de tester les requetes une par une à la main, il faut donc automatiser la vérification des requêtes. Pour cela, lors de la création des paysages à l'aide du logiciel R, Il est créé un dossier contenant les éléments suivants :

- un dossier par paysage contenant :
  - Le paysage au format .xml avec l'objectif pour l'état final inclus dans son système
  - 30 fichiers de requêtes (voit Parties résultats)
  - verifyta, l'exécutable d'Uppaal pour tester les requêtes

— un fichier .sh par paysage qui permet de tester chaque requête et d'enregistrer le résultat sous format .txt dans le répertoire Résultat
— un fichier .sh exécutnant l'ensemble des shell évoqué ci-dessus. Son exécution permet donc de tester toutes les requêtes sur tous les paysages.

Il suffira d'exécuter le fichier shell maître qui permettra d'adresser les requêtes à l'ensemble des dynamiques que nous voulons tester. Pour cela, ce fichier appelera les différents fichiers shell réalisant l'interrogation par verifyta des paysages, un fichier par paysage. Ce shell maître ne fait donc qu'appeler l'ensemble des autres fichiers shell, ce qui peut permettre de paralléliser le travail à l'aide de serveurs dédiés. Cette démarche est résumée dans la figure 12.

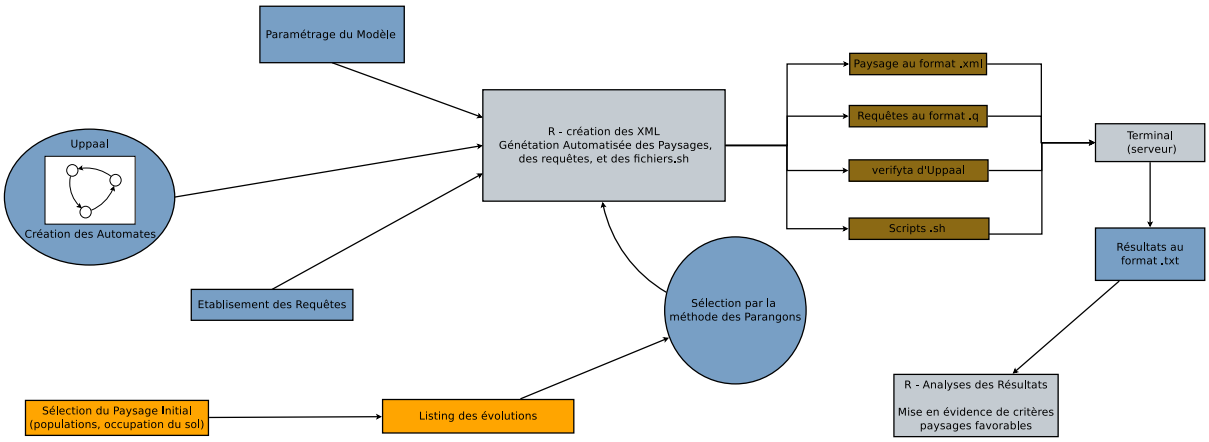


FIGURE 12 – Organisation pour l'automatisation de la vérification

Pour réaliser l'ensemble de ces exécutions, un serveur dédié de l'IRISA a été utilisé, Igrida, avec une mémoire vive de 50 Go pendant 24 heures (par rapport aux 8Go de l'ordinateur à disposition, ce qui est déjà très performant pour un ordinateur individuel). Ceci a pour but d'accélérer la vérification des requêtes et d'avoir des réponses au plus vite.

5.5 Analyse des résultats

A partir des résultats obtenus, nous allons pouvoir analyser les critères paysagers influençant les critères d'évolutions qui nous intéressent. Le résultat d'une requête étant vrai ou faux, nous allons réaliser des régressions logistiques pour rechercher l'influence des facteurs sur la vérification des requêtes. Nous étudierons à la fois les résultats pour les requêtes s'intéressant à l'existence d'un chemin, puis à celles portant sur l'implication, pour chaque critère d'évolution seul, puis au couple.

Pour les requêtes simples, c'est-à-dire avec un seul critère d'évolution, il s'agit d'une régression logistique. En utilisant des inégalités larges, il en résulte que si une requête est vérifiée pour une valeur donnée, alors elle l'est également pour des valeurs plus grandes. A partir donc des trois valeurs que nous avons testées, nous les regroupons en une seule variable à quatre modalités ordonnées, à savoir les trois valeurs et une pour les dynamiques ne vérifiant aucune requête. Pour des soucis de notation, on notera les modalités de 0 à 3, selon le niveau le plus restrictif vérifié, 3 correspondant à la plus faible valeur testée, et 0 à aucune requête vérifiée. Il s'agit donc du nombre de requêtes qui sont vraies pour une dynamique.

Nous allons donc réaliser des régressions logistiques sur les variables suivantes :

— $Pest_{Path}$: correspondant aux résultats des requêtes sur l'existence d'un chemin pour le critère portant sur les pesticides

— $Pest_{Imply}$: correspondant aux résultats des requêtes sur l'implication pour le critère portant sur les pesticides
— $Prey_{Path}$: correspondant aux résultats des requêtes sur l'existence d'un chemin pour le critère portant sur les nuisibles
— $Prey_{Imply}$: correspondant aux résultats des requêtes sur l'implication pour le critère portant sur les nuisibles

Nous allons rechercher les critères paysagers qui vont influencer les résultats des requêtes sur les deux critères :

$$P([Y = y|X]) = \frac{e^{f(X)}}{1 + e^{f(X)}} \tag{10}$$

avec :
— Y : une variable aléatoire portant sur le résultat des requêtes
— y : une modalité de Y, tel que $y \in Supp(Y)$
— X : l'ensemble des critères d'une dynamique paysagère
— $f(X)$: une fonction de X

L'objectif sera de rechercher les éléments de X qui permettront d'expliquer au mieux les variations des différentes variables. Pour cela, nous utiliserons une démarche pas-à-pas ascendante pour rechercher le meilleur modèle.

En ce qui concerne les requêtes portant sur les deux critères, les modalités restent ordonnées mais on cherche à prédire un couple de variable, dont les couples de modalités ne peuvent être ordonnées. Il sera donc possible de faire seulement une régression logistique non ordonnée, pour essayer de dégager des tendances en lien avec les modalités qui sont ordonnées pour chaque variable. Pour les modalités, on reprend le même principe que précédemment, en prenant les valeurs les plus faibles pour les deux critères où la requête est vérifiée. La régression logistique sera donc la forme suivante :

$$P([Pest = y] \cap [Prey = z]|X) = \frac{e^{f(X)}}{1 + e^{f(X)}} \tag{11}$$

avec
— $Pest$: une variable aléatoire portant sur le critère d'utilisations des pesticides
— y : une modalité de Y, tel que $y \in Supp(Pest)$
— $Prey$: une variable aléatoire portant sur le critère de présence des nuisibles
— z : une modalité de $Prey$, tel que $z \in Supp(Prey)$
— X : l'ensemble des critères d'une dynamique paysagère
— $f(X)$: une fonction de X

6 Résultats

6.1 Extraction des parangons

Nous avons donc généré les 2^{21} possibilités de dynamique paysagères. Parmi cet ensemble, seules celles correspondant aux critères de restriction ont été conservées, à savoir entre 40 et 65% de cultures et pas plus de 30% de différences (8 éléments), ce qui en laisse 76 714, soit moins de 4% de toutes les dynamiques possibles. Pour ces éléments conservés, les critères paysagers ont été calculés, et l'ACP a été réalisée. La projection des variables sur les deux premières dimensions est présentée en figure 13. Il est possible d'observer la forte inertie des deux premiers axes, qui recueillent plus de 60% de l'information à eux deux. Si l'on regarde le cumul sur les autres dimensions, 90% de l'information est résumé

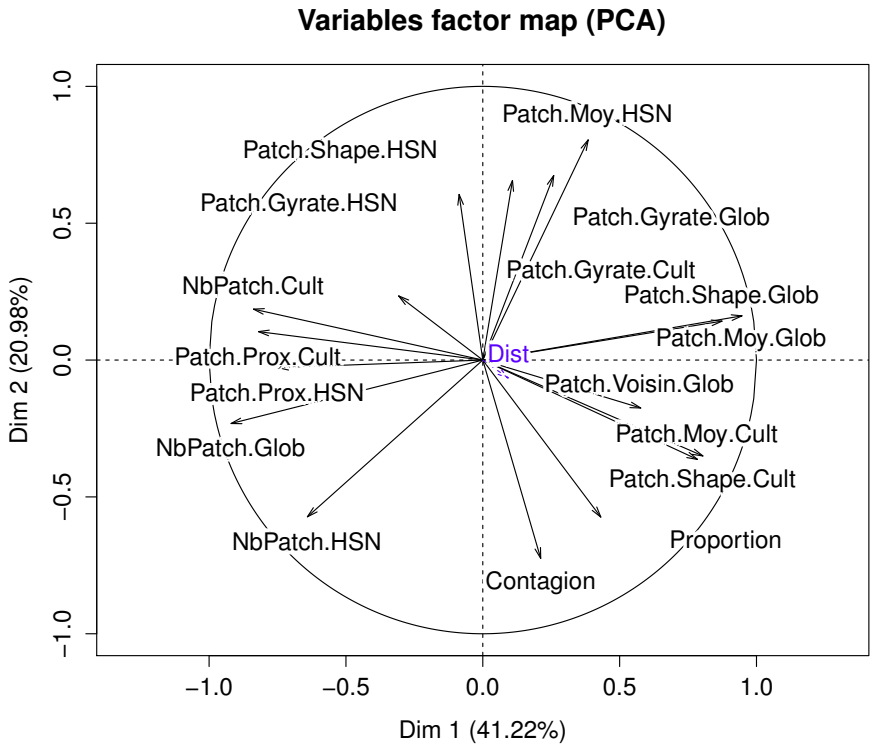


FIGURE 13 – Projection des variables sur les deux premières dimensions de l'ACP

avec les six premières dimensions sur les dix-sept existantes. Ces critères sont donc en partie redondants, avec certains critères très proches sur les trois premières dimensions, comme *Patch.Moy.Glob* et *Patch.Shape.Glob*, ou encore son pendant pour les cultures (*Patch.Moy.Cult* et *Patch.Shape.Cultlob*). Les valeurs propres, ainsi que les coordonnées et les cosinus des variables sur les huit premières dimensions sont présentées en Annexe 8. Cette ACP permet donc d'éliminer de la redondance dans l'information entre les critères descriptifs des dynamiques. Pour la variable illustrative, *Dist*, qui correspond à la distance au paysage initial, elle est très faiblement corrélée avec tous les axes issus de l'ACP. Les tendances des dynamiques paysagères ne semblent donc pas en lien avec la distance au paysage de référence.

Suite aux résultats de l'ACP, où toutes les variables ont été conservées pour ne pas perdre d'information, une classification ascendante hiérarchique a été réalisé, avec un nombre de clusters déterminé par le nombre de dynamiques paysagères que l'on s'autorise à tester, moins un, car dans tous les cas, la dynamique paysagère correspondant à l'état initial du paysage est testée. Dans notre cas, on s'autorise à tester 41 éléments, donc 40 groupes ont été créés, et le premier parangon a été extrait, ce qui permet d'avoir un échantillon le plus représentatif, avec des distances intra-groupes la plus faible possible. On récupère donc la description des dynamiques paysagères, c'est-à-dire l'occupation du sol désiré dans notre laps de temps.

Il faut à présent tester les dynamiques extraites à l'aide des requêtes élaborées, afin de trouver les critères les plus avantageux pour les différents critères d'évolutions.

6.2 Résultats des Requêtes

L'un des problèmes rencontrés lors de la vérification des requêtes est la limite d'utilisation en mémoire d'Uppal, et donc de verifyta par extension. En effet, le logiciel ne peut pas utiliser plus de 2 Go de mémoire pour tester les requêtes, la mémoire supplémentaire lui servant à stocker de façon temporaires ses résultats. L'utilisation d'un serveur dédié de 50 Go n'a donc servi à rien, la plus grande partie de la mémoire n'étant pas exploitée. Un ordinateur avec 8 Go de mémoire étant bien suffisant.
De plus, si toute la mémoire est utilisée au fur et à mesure de la vérification pour stocker les résultats intermédiaires, alors verifyta s'arrête sans donner de résultat, quelque soit le temps qui s'est écoulé. Nous sommes donc repassés sur un ordinateur classique, en diminuant la taille du lattice à 4*4, puis à 3*3.
Pour ces raisons, les simulations ont été lancées sur un ordinateur classique avec 8 Go de mémoire vive. Au bout d'une semaine, seulement la moitié des résultats pour le premier paysage avaient été obtenus, à savoir sur l'existence de chemins pour atteindre les objectifs. Les premières requêtes d'implications n'ont pas abouti suite à un manque de mémoire de l'ordinateur, et se sont donc arrêtées avant de donner des résultats. Il n'est donc pas possible de réaliser les régressions logistiques évoquées précédemment, n'ayant les résultats que pour un individu.

7 Discussion

7.1 Concernant la sélection de l'échantillon

L'utilisation de l'ACP a permis de mettre en évidence certains liens entre plusieurs variables. Il est donc intéressant de passer par cette étape de l'ACP avant la classification pour obtenir la meilleure représentation de toutes les dynamiques. Le résultat permet donc d'avoir une bonne représentation des dynamiques selon les critères paysagers. Si on avait utilisé la description des dynamiques par les éléments qu'on la compose, la donne aurait été faussée par des dynamiques qui peuvent être très éloignées d'un point de vue de la distance, mais très proche en terme d'organisation, comme ça serait le cas pour deux paysages symétriques.
Cependant, avec l'utilisation de la fonction *HCPC* du package *FactoMineR*, les résultats demandent beaucoup de mémoire. En effet, réaliser une classification sur 76 714 individus avec 18 variables (dont une illustrative) nécessite 21.9 Gb, ce qui est trop élevé pour R sur un ordinateur portable classique. Il a donc été nécessaire de réaliser la classification ascendante hiérarchique sans cette fonction pour ne pas être saturé en termes de mémoire, en passant par des étapes intermédiaires. Il serait donc nécessaire d'intégrer des éléments permettant de travailler sur des jeux de données de grandes dimensions, et ce dès la génération des paysages et leur caractérisation (qui peut prendre plus d'une journée selon les critères de sélection des dynamiques après leur génération).
Pour gagner du temps pour de futures simulations de taille plus importante, si Uppaal réussissait à devenir plus performant et la taille du système réduite, ce dont nous parlerons après, il pourrait être intéressant de rechercher le nombre de dimensions à conserver suite à l'ACP. En effet, en réduisant le nombre de dimensions utilisés sur la classification, on gagne en mémoire libre, ce qui a fait défaut à R, mais aussi en temps de calcul, sans pour autant perdre beaucoup d'informations, comme on peut le voir en Annexe 8. Il pourrait donc être intéressant de rechercher le nombre d'axes suffisants pour conserver un certain degré de l'inertie, 90% par exemple, en fonction de la taille du lattice.

7.2 Concernant les requêtes adressées

L'absence de résultats de la part des requêtes ne veut pas dire qu'il n'est pas possible de tirer de conclusion sur ce travail. En effet, il a été possible de voir que l'existence d'un chemin pour un

seul critère est facilement vérifiable, mais que l'implication reste compliquée à vérifier (temps de réponse trop long). Cette complexité vient de la part non-déterministe du modèle, ce qui est appelé aléatoire dans les modèles numériques classiques. En effet, comme dit précédemment, la partie non-déterministe permet par l'utilisation des requêtes d'intégrer toutes les possibilités au cours d'une seule simulation.
Contrairement aux requêtes portant sur l'existence avec un seul critère, celles avec les deux critères ont été plus compliquées à vérifier. Le temps de réponse et le nombre d'états testés étaient bien supérieurs à ce qui était nécessaire pour les requêtes à un critère. Certaines, les plus restrictives sur les critères d'évolutions, n'ont d'ailleurs pu être résolues par Uppaal, par manque de mémoire disponible.
Ce point montre que ces nouvelles approches en modélisation sont encore en développement. L'un des enjeux et de réussir à réduire la taille de l'espace de recherche pour Uppaal, qui dépend du produit du nombre de localisation de chaque automate par le nombre de liaisons possible pour chaque état du système. En ce qui concerne les liaisons, le problème se pose essentiellement lors du tirage aléatoire d'une valeur dans un intervalle, comme cela a été évoqué précédemment. Pour les automates, un des objectifs seraient de réduire le nombre d'automates, notamment en travaillant sur la dispersion, qui représente plus de la moitié des automates. Même si ces automates sont relativement de petites tailles, leur nombre joue de façon importante sur l'exploration des évolutions.

Conclusion

Après avoir vu les forces et les faiblesses de la modélisation qualitative, et plus particulièrement des automates temporisés, nous avons vu qu'il est possible de représenter des agro-écosystèmes à l'aide de ces approches qualitatives. La représentation de la compartimentation d'un paysage, mais également les flux de population en son sein ont été réalisées pour la première fois et permettent une simplification des populations représentées en ne s'intéressant pas à leurs fluctuations mineures. Il est donc possible de modéliser les évolutions spatiales par un modèle qualitatif sans pour autant trop dégrader le processus de représentation.
L'absence de résultats du de vue des critères d'évolution a permis de montrer qu'il était encore nécessaire de développer des outils plus performants en lien avec les automates temporisés, et qu'il est nécessaire de réfléchier à la compléxité d'un modèle lors de sa réalisation. En effet, plus un modèle sera complexe, plus les simulations nécessiteront des moyens importants. Il est donc nécessaire de faire des compromis entre l'exactitude du modèle et les moyens à disposition.
Il serait donc intéresser de pouvoir comparer les résultats entre approche numérique et approche qualitative, en comparant leurs forces et leurs faiblesses grâces aux résultats. Cela nécessite cependant des outils plus performants que ceux existants déjà pour des applications réelles.

Références

[1] Alexandre David, Tobias Amnell, and Martin Stigge. Uppaal Small Tutorial.

[2] Rajeev Alur. Timed Automata. In Nicolas Halbwachs and Doron Peled, editors, *Computer Aided Verification*, number 1633 in Lecture Notes in Computer Science, pages 8–22. Springer Berlin Heidelberg, 1999.

[3] Gerd Behrmann, Re David, and Kim G. Larsen. A tutorial on uppaal. pages 200–236. Springer, 2004.

[4] Patricia Bouyer. Weighted Timed Automata : Model-Checking and Games. *Electronic Notes in Theoretical Computer Science*, 158 :3–17, May 2006.

[5] Patricia Bouyer and François Laroussinie. Model Checking Timed Automata. In Stephan Merz and Nicolas Navet, editors, *Modeling and Verification of Real-Time Systems*, pages 111–140. ISTE, 2008.

[6] Bert Bredeweg and Paulo Salles. Qualitative models of ecological systems — Editorial introduction. *Ecological Informatics*, 4(5–6) :261–262, November 2009.

[7] European comission. Quantitative versus qualitative methods, 2005.

[8] Marie-Odile Cordier, Christine Largouët, and Yulong Zhao. Model-Checking an Ecosystem Model for Decision-Aid. In *IEEE 26th International Conference on Tools with Artificial Intelligence*, Limassol, Cyprus, November 2014.

[9] Jeffrey M. Dambacher, Hiram W. Li, and Philippe A. Rossignol. Qualitative predictions in model ecosystems. *Ecological Modelling*, 161(1–2) :79–93, March 2003.

[10] Christine Largouët, Marie-Odile Cordier, Yves-Marie Bozec, Yulong Zhao, and Guy Fontenelle. Use of timed automata and model-checking to explore scenarios on ecosystem models. *Environmental Modelling & Software*, 30 :123–138, April 2012.

[11] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Efficient timed model checking for discrete-time systems. *Theoretical Computer Science*, 353(1–3) :249–271, March 2006.

[12] Andre Botequilha Leitao, Joseph Miller, Jack Ahern, and Kevin McGarigal. *Measuring Landscapes : A Planner's Handbook*. Island Press.

[13] J. D. Murray, editor. *Mathematical Biology*, volume 18 of *Interdisciplinary Applied Mathematics*. Springer-Verlag, New York, 1993.

[14] Louise Trave-Massuyes, Liliana Ironi, and Philippe Dague. Mathematical Foundations of Qualitative Reasoning. *AI Magazine*, 24(4) :91, December 2003.

[15] Louise Travé-Massuyès, Francesc Prats, Mónica Sánchez, and Núria Agell. Relative and absolute order-of-magnitude models unified. *Annals of Mathematics and Artificial Intelligence*, 45(3-4) :323–341, December 2005.

[16] Shengsheng Wang and Dayou Liu. Knowledge representation and reasoning for qualitative spatial change. *Knowledge-Based Systems*, 30 :161–171, June 2012.

[17] Yulong Zhao. *Modélisation qualitative des agro-écosystèmes et aide à leur gestion par utilisation d'outils de model-checking*. phdthesis, Université Rennes 1, January 2014.

[18] Yulong Zhao, Christine Largouët, and Marie-Odile Cordier. EcoMata : Un logiciel d'aide à la décision pour améliorer la gestion des écosystèmes. *Ingénierie des Systèmes d'Information*, 16(3) :85–111, 2011.

Annexe 1 | Définitions formelles des automates temporisés

Définition d'un automate temporisé : un *automate temporel* est un n-uplet $\langle L, L^0, \sum, X, Inv, E \rangle$, [2], [5] où :

- L : ensemble de localisations
- $L^0 \subseteq L$: ensemble de localisations initiales
- $\sum$: ensemble d'actions
- X : ensemble d'horloges
- Inv : ensemble des conditions d'invariances de chaque état
- $E \subseteq L \times \sum \times 2^X \times \phi(X) \times L$: un ensemble de transitions entre états, avec $\phi(X)$ l'ensemble des contraintes temporelles.

Les contraintes pouvant s'appliquer sont celles liées à des comparaisons relatives. On note ν l'interprétation d'une horloge de X. Elle assigne à chaque horloge une valeur (rationnelle, entière ou réelle selon les automates). On dit que ν satisfait une contrainte temporelle $\varphi \in \phi$ si et seulement si φ est estimée vrai si elle est en accord avec les valeurs données par ν.

Définition d'un système de transition On modélise le système à l'aide de graphiques de transitions entre états. Un *système de transition* est un n-uplet $\langle L, L^0, \sum, \rightarrow \rangle$, avec :

- L : un ensemble d'états
- $L^0 \subseteq L$, un ensemble d'états initiaux
- $\sum$: un ensemble d'événements
- $E \subseteq Q \times \sum \times Q$: un ensemble de transitions. Un changement $\langle s, a, \varphi, \lambda, s' \rangle$ est une transition de s à s', sur un événement a. φ est une contrainte temporelle sur X et $\lambda \subseteq X$ donne l'ensemble des horloges à remettre à zéro.

On note S_A le système de transition associé à l'automate A

Définition d'une transition dans S_A On définit une *transition* $\langle l, a, l' \rangle$ de $\rightarrow$, allant de la l'état l à l', par l'événement a. On peut également l'écrire $l \xrightarrow{a} l'$. Un système complexe peut être défini comme un produit de système de transition. On définit un automate temporisé A en l'associant avec un système de transition S_A. Un état de S_A est une paire (s, ν) où s est une localisation de A, et ν est une horloge de X qui satisfait les invariants $Inv(s)$. Il existe deux types deux transitions dans S_A :

- une transition due à un écoulement de temps où la localisation de bouge pas. $\forall \delta \geqslant 0, (s, \nu) \xrightarrow{\delta} (s, \nu + \delta)$ si $\forall 0 \geqslant \delta' \geqslant \delta, \nu + \delta$ satisfait les invariants Inv(s)
- une transition due à un changement de localisation, où le temps ne change pas, mais certaines horloges peuvent êtres réinitialisées. Pour un état (s, ν) et un changement $\langle s, a, \varphi, \lambda, s' \rangle$ tel que ν satisfait φ, $(s, \nu) \xrightarrow{a} (s', \nu[\lambda := 0])$. $\nu[\lambda := 0])$ signifie que $\nu(x) = 0$ si $x \in \lambda$. De façon plus simple, si λ est l'ensemble des horloges de X qui sont réinitialisées suite à la transition.

Un seul type des deux types de transitions évoquées peut se dérouler à un même instant, même si plusieurs transitions d'un même type peuvent se dérouler en même temps. Par exemple, toutes les horloges incrémentent en même temps, avec le principe qu'elles s'écoulent à la même vitesse. (Si une horloge augmente d'une seconde, alors toutes les autres horloges augmentent aussi d'une seconde). Sinon, plusieurs évolutions de localisations entre plusieurs automates peuvent aussi avoir lieu à un même instant (du point de vue temporel).

Définition d'un Produit synchronisé On définit par *Produit synchronisé* le produit de deux automates temporisés [2]. Soient $A_1 = \langle L_1, L_1^0, \sum_1, X_1, Inv_1, E_1 \rangle$ et $A_2 = \langle L_2, L_2^0, \sum_2, X_2, Inv_2, E_2 \rangle$ deux automates temporisés, avec deux ensembles d'horloges disjoints $X_1 \cap X_2 = \emptyset$. Alors le produit $A_1 \| A_2$ est un automate temporisé $\langle L_1 \times L_2, L_1^0 \times L_2^0, \sum_1 \cup \sum_2, X_1 \cup X_2, Inv, E \rangle$, où $Inv(s_1, s_2) = Inv(s_1) \wedge Inv(s_2)$ et les horloges sont définies par :

— for $a \in \sum_1 \cap \sum_2, \forall \langle s_1, a, \varphi_1, lambda_1, s_1' \rangle \in \sum_1$ et $\forall \langle s_2, a, \varphi_2, lambda_2, s_2' \rangle \in \sum_2$, E contient $\langle (s_1, s_2), a, \varphi_1 \wedge \varphi_2, \lambda_1 \cup \lambda_2, (s_1', s_2') \rangle$

— Si $a \in \sum_1 \setminus \sum_2, \forall \langle s, a, \varphi, \lambda, s' \rangle \in E_1$ et $\forall \in L_2$, alors E contient $\langle (s,t), a, \varphi, \lambda, (s', t) \rangle$

— Si $a \in \sum_2 \setminus \sum_1, \forall \langle s, a, \varphi, \lambda, s' \rangle \in E_2$ et $\forall \in L_1$, alors E contient $\langle (t,s), a, \varphi, \lambda, (t, s') \rangle$

Annexe 2 | Utilisation d'Uppaal

Uppaal 4.0 est un outil développé par les universités d'Uppsala (Suède) et d'Aalborf (Danemark) [1], [3] (D'où le nom qui est la fusion du début des deux noms). Il permet la synthèse d'automates temporisés et la synchronisation entre plusieurs d'entre eux. Pour cela, il s'appuie sur les définitions évoquées précédemment sur un (ou plusieurs) modèle(s) et le(les) système(s) de transition(s) associé(s). Une fois les automates créés, il permet d'utiliser des outils de model-checking pour tester certaines conditions sur les événements possibles au cours de simulation.
On distingue donc deux parties :une première de création des automates et leurs synchronisations (Figure 14), accompagnée de la vérification des syntaxes et de la possibilité de tester l'évolution du système ; une seconde pour le model-checking permettant de tester les contraintes désirée Nous allons rapidement évoquer ces deux parties et leur mode de fonctionnement propre.

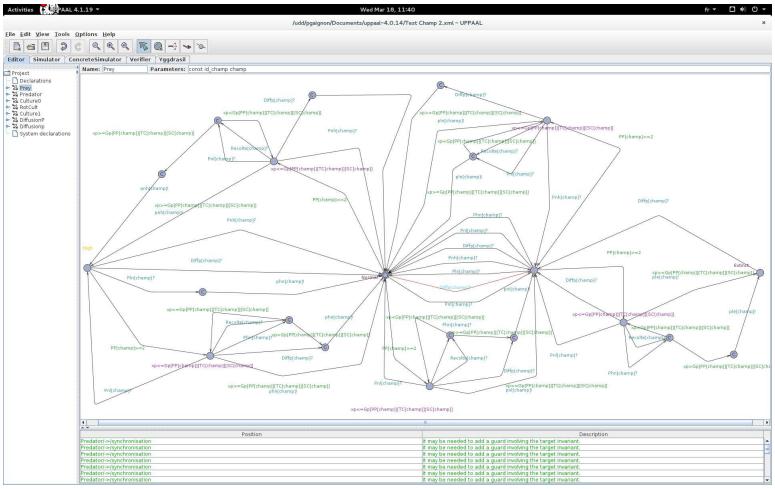


FIGURE 14 – Interface d'Uppaal pour les automates

En ce qui concerne les créateurs d'automates, elle utilise des *templates* qui correspondent aux automates. Pour chaque *template*, on définit deux types d'objets principaux :

— Location : il s'agit des localisations du systèmes. Il est possible :

initial : localisation à partir de laquelle le *template* sera lancé pour les simulations.

urgent : quand une telle localisation est atteinte, le temps ne peut plus s'écouler tant que l'automate reste dans cet état. Il est cependant possible que d'autres automates évoluent avant que cette localisation ne soit quittée.

commited : un peu comme l'état urgent, le temps reste bloqué tant que l'automate reste à cette localisation. Cependant, seules d'autres automates étant aussi sur des localisations *committed* peuvent évoluer, il est plus restrictif que l'état *urgent*. On limite donc l'évolution à certains cas précis (qu'on peut qualifier de prioritaires).

Pour chaque localisation on peut également établir les invariants, conditions qui quand elles sont ne respectées pas, obligent l'évolution du système.

— Edge : ils permettent les transitions entre les états du système selon les contraintes établies dessus. On distingue quatre éléments pour chaque liaison :

select : Permet d'attribuer une valeur aléatoire à une variable.

guard : Ensemble des conditions à vérifier pour que cette transition ait lieu

Synchr : permet la synchronisation entre deux ou plus automates pour des évolutions simultanées. Pour cela, il faut créer une action (a) avec un émetteur(a !) et un receveur(a ?) ou plusieurs (selon la nature de la synchronisation).

Update : lors du passage de par cette liaison, elle permet de modifier les valeurs de certaines variables (comme la mise à zéro des horloges par exemple)

A partir de ces outils, on peut créer un ou plusieurs automates, avec des variables propres à chaque automate ou globales. On a donc une forme de "super-automate" qui s'occupe de gérer tous les autres automates (les *templates*) et de les faire communiquer entre eux.

A l'aide de l'onglet simulation, il est possible de tester le "super-automate" créé (avec une vérification de la syntaxe préalable). Cela permet de tester les évolutions que peut prendre le modèle à partir des contraintes qu'il possède et vérifier que certaines évolutions existent bien. On peut y choisir soi-même les évolutions possibles, ou bien laisser le logiciel générer aléatoirement un chemin en suivant l'évolution au fur et à mesure. Enfin, à l'aide du dernier onglet, il est possible d'y réaliser du model-checking à l'aide de requête TCTL. Une grammaire adaptée y est développée mais elle permet toutes les requêtes classiques de ce genre de logique.

Il existe plusieurs extensions à Uppaal pour des besoins spécifiques qui n'existent pas forcément dans la version d'origine. C'est le cas d'Uppaal-Tiga qui permet de générer des automates de façon automatique à partir de modèles, ce qui évite de devoir tous les créer manuellement, ce qui peut être assez coûteux en terme de temps. C'est notamment ce qui a été utilisé dans les deux exemples développés ci-dessous, comme lien entre les interfaces développés et Uppaal.

Annexe 3 | Paramétrage du modèle

Pour permettre le lancement des simulations, il est nécessaire de réaliser le paramétrage du modèle. L'ensemble des variables qui doivent être paramétrer sont présentées ici, à l'exception de celles qui ne peuvent être modifiées, comme les horloges, et les chan. Pour les vecteur et les tableaux multi-dimensionnels, les éléments entre parenthèses indiquent la taille de la dimension correspondante. De plus, la plupart de ces variables sont les mêmes quelque soit la dynamique pa

N : nombre d'éléments dans le paysage à l'état initial

PP(N) : état de la population de prédateur pour les N éléments à l'état initial

pp(N) : état de la population de proie pour les N éléments (pour les matrices, la population est existante mais doit être définie) à l'état initial

TS(N) : Indique l'occupation du sol pour les éléments du paysage à l'état initial

RC(N) : Indique la rotation culturale en cours si c'est une culture, ou l'occupation du sol sinon, à l'état initial

RCObj(N) : dynamique paysagère que l'on souhaite tester (objectif final pour l'état du paysage). Cette variable est différente pour chaque dynamique.

NbRotation : nombre de rotation culturale possible dans le paysage

NbCult(NbRotation) : nombre de culture pour chaque rotation

RotRec(NbRotatio)(NbCult) : Date de récolte de chaque culture dans chaque rotation

RotSem(NbRotation)(NbCult) : Date de semis de chaque culture dans chaque rotation

TempsRot(NbRotation) : temps total de chaque rotation culturale

GP(liaison)(TS)(pp) : indique le temps pour les changements d'états des prédateurs selon l'évolution, l'occupation du sol et l'état de la population de proie

Gp(liaison)(TS)(pp) : indique le temps pour les changements d'états des proies selon l'évolution, l'occupation du sol et l'état de la population de prédateur

TabPrior(TS_i)(PP_i)(TS_j)(PP_j) : Priorité sur les migrations selon les occupations du sol et les états des populations entre les deux élément i et j qui cont contigus.

VitPrior(TS_i)(PP_i)(TS_j)(PP_j) : Vitesse de migration selon les occupations du sol et les états des populations entre les deux élément i et j qui cont contigus.

Diffusion(pp_e)(pp_r)(dist) : probabilité qu'il y ait une dispersion de proie selon la population émettrice (pp_e), réceptrice (pp_r) et la distance entre les deux éléments.

Annexe 4 | Fonction de gestion des rotations culturales

A partir de deux fonctions réalisées sous Uppaal, à l'aide d'un language s'approchant du C++, les dates de cultures et de semis prévues pour certaines rotations peuvent être récupérées pour gérer l'alternance programmée des cultures. A noter que le temps total de rotation doit correspondre à un écoulement d'années entières, et donc inclure les temps de jachère.

Recherche des dates de semi : Test

```
//Fonction pour determiner debut de Recolte
int DebCult(int Temps, int Rotation, int RotSem[NbRotation][NbCult],
            int TempsRot[NbRotation], int NbCult)
{
    int i=0;
    int tps=Temps; //Horloge Globale du systeme sauvegardee
    int DateDebut=TempsRot[Rotation];
    int ajout=0;
    // On va revenir dans un intervalle de temps
    // correspondant a celui de la rotation
    while (TempsRot[Rotation]<=tps)
    {
        tps=tps-TempsRot[Rotation];
        ajout=ajout+1;
    }
    // On va chercher quelle culture on va semer
    while (i<NbCult)
    {
        if (tps<=RotSem[Rotation][i])
        {
            DateDebut=RotSem[Rotation][i];
            i=NbCult;
        }
        i=i+1;
    }
    DateDebut=DateDebut+ajout*TempsRot[Rotation];
    return DateDebut ;
}
```

Recherche des dates de récolte : On a semé un champ, on recherche la date de récolte qui lui est attribuée pour établir les contraintes temporelles nécessaires. Ci-dessous se trouve le code utilisable pour cela.

```
//Fonction pour determiner fin de Recolte
int FinCult(int Temps, int Rotation, int RotRec[NbRotation][NbCult],
            int TempsRot[NbRotation], int NbCult)
{
    int i=0;
    int tps=Temps;
    int DateRec=TempsRot[Rotation];
    int ajout=0;
    // On va revenir dans un interval de temps
    // correspondant[U+FFFD]celui de la rotation
    while (TempsRot[Rotation]<=tps)
    {
        tps=tps-TempsRot[Rotation];
        ajout=ajout+1;
    }
    // On va chercher quelle culture on va r[U+FFFD]colter
    while (i<NbCult)
    {
        if (tps<=RotRec[Rotation][i])
        {
            DateRec=RotRec[Rotation][i];
            i=NbCult;
        }
        i=i+1;

    }
    DateRec=DateRec+ajout*TempsRot[Rotation];
    return DateRec;
}
```

Annexe 5 | Règles pour les Migrations

Règles pour qu'une migration soit possible :

1. Il n'est pas possible d'atteindre les états High et Normal dans une matrice
2. Pas de migration quand les deux populations sont à l'état Extinct ou High
3. Excepté pour la matrice, il faut deux niveaux de différence sur les états de population pour qu'il y ait une migration pour deux mêmes occupations du sol.
4. L'occupation des cultures est prioritaire sur toutes les autres. Il peut y avoir diffusion à partir du moment où la population de culture est au moins un niveau en dessous d'une population avec une autre occupation du sol.
5. L'occupation du sol nu est équivalente à celle de l'habitat semi-naturel d'un point de vue des migrations. Une migration HSN-HSN est équivalente à une migration Sol Nu-HSN
6. La migration d'une culture vers un sol nu ou HSN peut se faire s'il y au moins deux niveaux de populations de plus pour la population sur culture.
7. Entre deux matrices, la diffusion peut avoir lieu si les niveaux sont différents E/L donne L/E pour simuler le déplacement à travers les matrices.
8. Diffusion vers la matrice seulement quand on est à l'état High

Règles de Priorité entre les migrations :

1. On cherche à avoir les populations les plus élévés sur les cultures
2. Les diffusions inter-cultures sont toujours prioritaires
3. Plus l'écart entre populations est important, plus la migration est prioritaire
4. Même priorité entre Sol nu et HSN pour de même situation de population
5. La matrice diffuce vers l'extérieur facilement, mais il est difficile d'y faire venir des individus
6. Les diffusions inter-matrices ne se font que quand aucune autre n'est possible

Annexe 6 | Ensemble des Synchronisations présentes dans le système

Liste des chan : Un chan est une communication entre un émetteur, et un seul récepteur. L'émetteur ne peut réaliser sa transition s'il n'y a pas de récepteur

Pest : Action des Pesticides qui a forcément un impact sur les populations de proes

Semaison : Le champ vient d'être semé, ce qui induit un changement dans l'occupation du sol

DiffB : émis par un automate de migration d'un automate de prédateur, pour lui faire baisser son niveau de population.

DiffH : émis par un automate de migration d'un automate de prédateur, pour lui faire augmenter son niveau de population.

Liste des Broadcasts : A la différence des chan simples, les broadcasts chan peuvent émettre à plusieurs récepteurs à partir d'une seule émission, mais aussi évoluer sans qu'il y ait de récepteur. L'évolution de l'émetteur n'est donc pas entravé par l'absence de récepteur.

Recolte : Indique que la récolte a été effectué, il y a évolution du terrain

Terrain : Indique qu'il y a eu récolte/semis,

Evol : Indique une évolution de la population de prédateurs

Vent : Indique une dispersion aléatoire par le vent (phénomène aléatoire)

PriorP : Indique la présence d'une priorité dans les migrations, pouvant arrêter des migrations moins priorités

Diffr : Indique qu'un automate de proie va devoir faire évoluer son état de population suite à une dispersion de proie

Les broadcasts sur les évolutions de population : Ces chan indiquent des évolutions sur les niveaux de populations des espèces. Ils sont tous basés sur une même règle de construction, et sont tous codés à l'aide de trois lettres :

— la première indique si l'émetteur est un prédateur, avec un P majuscule, ou une proie, avec un p minuscule.

— La seconde indique le niveau où se trouvait précédemment l'espèce, (h pour *high*, n pour *normal*, l pour *low* et e pour *extinct*).

— La troisième et dernière indique à quel état se trouve désormais l'espèce concernée.

En voici la liste complète : Pnl, Pln, Pnh, Phn, Ple, pnl, pln, pnh, phn, ple.

Annexe 7 | Définition des critères Paysagers

Cette annexe présente l'ensemble des indicateurs des dynamiques paysagères pour les caractériser. Ces indicateurs sont spécifiques à des paysages de type lattice. Certains critères sont servis pour caractériser plusieurs occupations du sol. Ils sont repérés de la manière suivante :

— .Env : le critère indique en trois exemplaies, un pour toutes les occupations du sol, un pour les cultures et un pour les HSN.
— .Spe : le critère ne peut caractériser qu'une occupation du sol à la fois. Il existe donc deux fois, une pour les cultures et une autre pour les HSN.

Dist : Distance au paysage de référence, c'est-à-dire le nombre d'éléments ayant des occupations du sol qui différent. Ses valeurs sont entières, situées entre 0 et une Valeur seuille fixée

Proportion : Pourcentage de culture sur le paysage, la quantité de matrice étant fixe car elles sont inamovibles. Il permet de décrire la répartition des occupations du sol

Nb.Patch.Env : Nombre de patchs présents dans le paysage

Patch.Moy.Env : Taille moyenne des patchs présents dans le

Patch.Shape.Env : Indicateur de la complexité géométrique des patchs. Il s'agit d'un ratio entre le périmètre du patch et celui pour le patch le plus simple (périmètre minimal pour le nombre donné d'éléments dans le patch) :

$$Patch.Shape.Env = \frac{p_i}{min(p_i)}$$

avec :
— p_i : périmètre du patch i
— $min(p_i)$: périmètre minimal pour un patch de même taille que le patch i, qui se calcule à partir de a_i, taille du patch i et n, partie entière de la racine carrée de a_i :
  — $min(p_i) = 4n$, si $n^2 = a_i$
  — $min(p_i) = 4n+2$, si $n^2 < a_i \leq n(n+1)$
  — $min(p_i)$, si $a_i > n(n+1)$

Patch.Gyrate.Env : Il s'agit de la distance moyenne entre cellules de lattice appartenant à un même patch :

$$Patch.Gyrate.Env = \sum_{r=1}^{R} \sum_{(i,j)\in r} \frac{b_{ijr}}{z_r}$$

avec

R : ensemble des patchs appartenant à la catégorie Env

b_{ijr} : distance de la cellule appartenant au patch r, située en (i,j), au centroïde du patch (location moyenne des cellules du patch)

z_r : nombre de cellulues dans le patch r

Contagion : quantification du degré avec le lequel les occupations du sol sont aggrégées, en opposition au fait d'être réparties en petit fragment. Elle se calcule à l'aide d'une matrice de proximité (comptabilisant les cellules adjacentes).

$$Contagion = (1 + \frac{\sum_{i=1}^{m}\sum_{k=1}^{m}[(P_i)(P_{ik})]ln[(P_i)(P_{ik})]}{C_{max}}) * 100$$

avec :
— m : nombre d'occupation du sol dans le paysage

— P_i : Pourcentage du lattice avec une occupation du sol de type i
— P_{ik} : Nombre de voisinage entre des cellules i et des cellues k, sur le nombre total de voisinage pour les cellules de type i.
— $C_{max} = 2ln(m)$

Patch.Voisin.Env : Distance moyenne au plus proche voisin de même occupation du sol :

$$Patch.Voisin.Glob = \frac{\sum_{i=1}^{n} b_i}{n}$$

avec :
— n : ensemble des patchs de type Env
— b_i : distance minimale à un patch de même nature pour le patch i

Patch.Prox.Spe : L'index de proximité est une mesure sans unité de l'isolation des patchs.

$$Patch.Prox.Spe = \sum_{r=1}^{n} \sum_{n}^{i} \frac{a_i}{b_{ir}^2}$$

avec
— a_i : taille du patch i
— b_{ir}^2 : distance entre les patch i et r

Annexe 8 | Résultats de l'ACP

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7	Dim.8
Proportion	0,4312	-0,5739	0,4836	0,3779	0,0494	-0,2025	0,0695	0,0379
NbPatch.Glob	-0,9198	-0,2318	-0,0004	-0,0728	0,0629	0,1607	0,2199	-0,0706
Patch.Moy.Glob	0,9472	0,1616	0,0467	0,0211	0,0438	0,0238	-0,0722	0,0548
Patch.Shape.Glob	0,8743	0,1423	0,0373	-0,1099	-0,1397	0,1189	0,1901	0,0716
Patch.Gyrate.Glob	-0,0865	0,6053	-0,2966	0,7092	0,0824	0,0595	0,1460	-0,0040
Contagion	0,2116	-0,7245	0,4127	0,2219	0,1461	0,0916	0,2656	0,2194
Patch.Voisin.Glob	0,5772	-0,1752	0,0523	-0,0935	0,5461	0,5026	-0,2579	-0,0302
NbPatch.HSN	-0,6409	-0,5728	-0,3750	-0,0585	-0,0355	0,2090	0,0517	0,1968
Patch.Moy.HSN	0,3851	0,8045	0,2609	-0,2081	-0,0317	0,0917	0,0411	-0,0454
Patch.Shape.HSN	0,2592	0,6743	0,3853	-0,2688	-0,1967	0,2886	0,2284	0,1534
Patch.Gyrate.HSN	0,1085	0,6560	-0,4548	0,1435	0,4699	-0,2193	0,2028	0,1301
Patch.Prox.HSN	-0,7486	-0,0276	-0,4921	-0,2006	-0,0978	0,2094	0,0783	0,0656
NbPatch.Cult	-0,8379	0,1861	0,3625	-0,0775	0,1887	0,0861	0,1618	-0,1826
Patch.Moy.Cult	0,7845	-0,3631	-0,2815	0,0840	-0,0658	0,0788	0,2305	-0,2292
Patch.Shape.Cult	0,8043	-0,3504	-0,3147	0,0060	-0,0537	0,1262	0,2122	-0,2056
Patch.Gyrate.Cult	-0,3075	0,2344	0,1045	0,7938	-0,2572	0,3374	-0,1600	-0,0160
Patch.Prox.Cult	-0,8195	0,1039	0,4622	0,0370	0,1606	-0,0260	0,1107	-0,1234

TABLEAU 2 – Coordonnees des variables sur les 8 premieres dimensions de l'ACP

Axe	Valeur propre	Pourcentage de variance	Variance cumulée (en %)
comp 1	7,0078	41,2226	41,2226
comp 2	3,5674	20,9846	62,2072
comp 3	1,8387	10,8158	73,0231
comp 4	1,5459	9,0938	82,1168
comp 5	0,7604	4,4728	86,5896
comp 6	0,7174	4,2202	90,8098
comp 7	0,5188	3,0518	93,8616
comp 8	0,2928	1,7225	95,5841
comp 9	0,2568	1,5105	97,0946
comp 10	0,2091	1,2300	98,3246
comp 11	0,1184	0,6964	99,0210
comp 12	0,0595	0,3502	99,3712
comp 13	0,0526	0,3091	99,6803
comp 14	0,0358	0,2107	99,8910
comp 15	0,0129	0,0758	99,9668
comp 16	0,0032	0,0189	99,9857
comp 17	0,0024	0,0143	100,0000

TABLEAU 3 – Valeurs propres et inertie sur les axes de l'ACP

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7	Dim.8
Proportion	0,1860	0,3293	0,2338	0,1428	0,0024	0,0410	0,0048	0,0014
NbPatch.Glob	0,8460	0,0537	0,0000	0,0053	0,0040	0,0258	0,0483	0,0050
Patch.Moy.Glob	0,8972	0,0261	0,0022	0,0004	0,0019	0,0006	0,0052	0,0030
Patch.Shape.Glob	0,7643	0,0202	0,0014	0,0121	0,0195	0,0141	0,0361	0,0051
Patch.Gyrate.Glob	0,0075	0,3663	0,0879	0,5029	0,0068	0,0035	0,0213	0,0000
Contagion	0,0448	0,5249	0,1703	0,0493	0,0214	0,0084	0,0705	0,0481
Patch.Voisin.Glob	0,3331	0,0307	0,0027	0,0087	0,2983	0,2526	0,0665	0,0009
NbPatch.HSN	0,4107	0,3280	0,1406	0,0034	0,0013	0,0437	0,0027	0,0387
Patch.Moy.HSN	0,1483	0,6473	0,0681	0,0433	0,0010	0,0084	0,0017	0,0021
Patch.Shape.HSN	0,0672	0,4546	0,1484	0,0723	0,0387	0,0833	0,0522	0,0235
Patch.Gyrate.HSN	0,0118	0,4303	0,2068	0,0206	0,2208	0,0481	0,0411	0,0169
Patch.Prox.HSN	0,5605	0,0008	0,2421	0,0402	0,0096	0,0438	0,0061	0,0043
NbPatch.Cult	0,7020	0,0346	0,1314	0,0060	0,0356	0,0074	0,0262	0,0333
Patch.Moy.Cult	0,6154	0,1318	0,0793	0,0070	0,0043	0,0062	0,0531	0,0525
Patch.Shape.Cult	0,6470	0,1228	0,0990	0,0000	0,0029	0,0159	0,0450	0,0423
Patch.Gyrate.Cult	0,0946	0,0550	0,0109	0,6301	0,0662	0,1139	0,0256	0,0003
Patch.Prox.Cult	0,6716	0,0108	0,2136	0,0014	0,0258	0,0007	0,0123	0,0152

TABLEAU 4 – Cosinus carré des variables sur les huits premières dimensions de l'ACP

AGRO CAMPUS OUEST	Diplôme : Ingénieur Agronome Spécialité : Statistiques appliquées Spécialisation / option : Enseignant référent : Julie JOSSE

Auteur(s) : Pierre GAIGNON Date de naissance : 24/01/1991	Organisme d'accueil : IRISA Adresse : 263 Avenue Général Leclerc, 35000 Rennes
Nb pages : 24 Annexe(s) : 8	
Année de soutenance : 2015	Maître de stage : Christine LARGOUET

Titre français : Modélisation et exploration d'un écosystème à dynamique spatiale par des outils de model-checking.

Titre anglais : Modelling and exploration of an ecosystem with spatial dynamic by model-checking

Résumé (1600 caractères maximum) :

La représentation de larges écosystèmes pose le problème des moyens nécessaires pour les modéliser. Pour cela, de nouvelles approches se développent pour pallier à ce problème. Ce rapport propose une nouvelle approche qualitative pour représenter des populations au sein d'un paysage soumis à l'action humaine. L'innovation vient de la modélisation des flux de population entre différents éléments paysagers en utilisant des automates temporisés, une méthode de modélisation qualitative. Il a ainsi été possible de réaliser une méthode partant de la modélisation du paysage jusqu'à la génération de ses évolutions possibles, et l'analyse des facteurs favorisant les critères agronomiques et écologies étudiés. Cependant, les outils à disposition restent limités et ne permettent pas encore d'application réelle.

Abstract (1600 caractères maximum) :

The representation of large ecosystem is problematic, because of necessary needs to model it. New approaches are developed to answer this problem. This report offers à new qualitative approach to represent populations inside a landscape, which is evolving under human actions. The innovation is te modelling of population flows with timed automata, a qualitative modelling method. It has been possible to create a method, which starts from the modelling of a landscape and ends at the creation of its possible development and the analysis of the factors promoting agronomic and ecological criteria. However, the actual tools are still limited and do not permit real applications.

Mots-clés : Automates temporisés, écosystèmes, modélisation

Key Words: timed automata, ecosystems, modelling