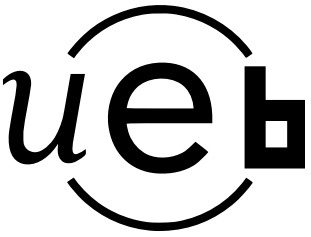
Modélisation qualitative des agro-écosystèmes et aide à leur gestion par utilisation d'outils de model-checking

Yulong Zhao

N° d'ordre : 00000 ANNÉE 2014



THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale SDLM

présentée par

Yulong ZHAO

Préparée à l'unité de recherche IRISA - UMR6074
Institut de Recherche en Informatique et Système Aléatoires
Composante Universitaire ISTIC

Modélisation qualitative des agro-écosystèmes et aide à leur gestion par utilisation d'outils de model-checking

Thèse soutenue à Rennes
le 13 janvier 2014

devant le jury composé de :

Jean-Marc FAURE
Professeur, ENS Cachan / *Rapporteur*
Frederick GARCIA
Directeur de recherche, INRA Toulouse / *Rapporteur*
René BAUMONT
Directeur de recherche, INRA Clermont-Ferrand / *Examinateur*
Bertrand BRAUNSCHWEIG
Directeur du centre INRIA, Rennes / *Examinateur*
Philippe FAVERDIN
Directeur de recherche, INRA St-Gilles / *Examinateur*
Marie-Odile CORDIER
Professeure, Université de Rennes 1 / *Directrice de thèse*
Chantal GASCUEL-ODOUX
Directrice de recherche, INRA Rennes / *Co-directrice de thèse*

Remerciements

Mes remerciements s'adressent particulièrement aux Mme Marie-Odile Cordier, Professeure de l'Université de Rennes 1, Mme Chantal Gascuel-Odoux, directrice de recherche de l'INRA Rennes, Mme Christine Largouët, Maître de conférence de l'AGROCAMPUS OUEST de Rennes pour m'avoir accordé cette opportunité précieuse dans ma vie.

Je remercie toutes les personnes avec qui j'ai travaillé pour m'avoir permis d'avancer dans mon travail.

Je tiens à remercier tous les membres de jury d'avoir accepté et de m'avoir fait l'honneur de juger mon travail de thèse.

Je remercie tous les membres (rêveurs) de l'équipe DREAM avec qui j'ai passé des moments très agréables.

Je remercie ma mère pour le support pendant la durée de ma thèse.

Table des matières

Première partie

Introduction et état de l'art

Chapitre 1

Introduction générale

Contexte

Les problématiques environnementales deviennent de plus en plus importantes de nos jours. Elles portent souvent sur des systèmes complexes qui possèdent la particularité de comporter de nombreuses entités interagissant. Par exemple un écosystème halieutique composé de plusieurs espèces de poissons, certaines étant des proies, d'autres étant des prédateurs, et dont l'évolution est "perturbée" par l'activité (contrôlable) de la pêche, essentielle pour la survie économique des acteurs concernés, mais aussi par l'occurrence (non contrôlable) de phénomènes climatiques tels que des forts vents ou l'augmentation de la température de l'eau. Un autre exemple d'agro-écosystème est celui de la gestion du pâturage dans une exploitation agricole centrée sur l'élevage laitier, où les besoins en nourriture des troupeaux doivent être satisfaits au mieux (par broutage de l'herbe ou consommation de fourrage), tout en tenant compte de l'impact environnemental de la fertilisation des prairies, en particulier en terme de transfert de nitrates.

Il est nécessaire de pouvoir analyser ces situations complexes afin de prendre les décisions les plus pertinentes, par exemple en terme de protection des espèces menacées dans le cas de l'halieutique, et de protection de la qualité des eaux dans le cas de la gestion de pâturage. Les analyses de ces situations complexes ont motivé le développement de modèles de simulation, qui sont souvent numériques, et de plus en plus complexes. Malheureusement les données pour construire de tels modèles sont difficiles à acquérir. Les connaissances sur le domaine étudié sont souvent incomplètes. Ces constatations ont amené à développer des modèles qualitatifs avec leurs avantages vis à vis d'un utilisateur. La simulation qualitative a été utilisée de manière satisfaisante dans de nombreux domaines liés à l'environnement tels que la physiologie végétale [RP97, VTZ+10], l'écologie terrestre [SBA06] et aquatique [TN06, GD01].

Dans le cas des applications réelles, la trop grande complexité des systèmes rend l'utilisation des modèles en simulation difficile : temps de réponse trop long pour une interaction avec l'utilisateur, nombre de données obtenues par simulation trop important (avalanche de données), ce qui rend nécessaire le développement de nouvelles approches.

Une première alternative consiste à proposer des outils de stockage et d'analyse des résultats de simulation de ces modèles afin de permettre une interaction avec l'utilisateur plus facile et adaptée [LTC+12, BCQG11]. On peut aller plus loin et proposer des outils souvent issus du domaine de l'intelligence artificielle pour extraire des connaissances à partir des données issues de la simulation ou de l'observation. C'est le cas du logiciel SACADEAU qui considère les résultats obtenus par simulation du modèle comme des exemples sur lesquels deux algorithmes d'apprentissage ont été utilisés pour apprendre des règles reliant conditions d'application des herbicides et degré de contamination des eaux [CGG+05]. Il est aussi possible de coupler ces simulateurs avec des outils proposant des stratégies optimales de gestion du système comme le fait MODERATO [BDD+01] et SEPATOU [CGMC99].

L'approche sur laquelle nous nous appuyons consiste à représenter le système étudié comme un système à événements discrets [AMBD98, AD94]. Cette représentation est en général à un niveau d'abstraction plus élevée que ce qui est fait dans les modèles numériques, mais est bien adaptée au cas où la dynamique du système est principalement liée à des interactions entre les entités concernés. C'est typiquement le cas d'un écosystème halieutique où les différentes espèces interagissent en tant que proie-prédateur, et où la gestion de pêche et les conditions environnementales peuvent aussi être vues comme des "prédateurs" impactant la biomasse des espèces. C'est ce que nous montrerons dans la partie consacrée au projet ECOMATA, un outil d'aide à la décision pour la gestion des écosystèmes. C'est aussi le cas dans le cas de la gestion du pâturage pour lequel l'alimentation des troupeaux et l'herbe poussant dans les prairies ont une interaction concurrente qui peut être modélisée en termes de systèmes à événements discrets. Dans ce dernier cas, comme on le verra dans la deuxième partie de cette thèse avec PATURMATA, la situation est un peu plus complexe puisqu'il faut tenir compte de la dynamique propre de la croissance de l'herbe et passer à une modélisation hybride.

Objectifs

Ayant ainsi représenté le système d'étude sous forme de systèmes à événements discrets, l'avantage est qu'il est possible de profiter du développement des techniques de model-checking qui ont été proposés à l'origine pour analyser des systèmes temps réel complexes, souvent informatiques ou industriels. Ces outils permettent de vérifier des propriétés (atteignabilité, sûreté) sur les systèmes sans simuler de manière exhaustive toutes les trajectoires d'évolution possibles, et ceci de manière efficace et symbolique [CGP02]. Nous proposons d'utiliser ces outils pour analyser les modèles obtenus. C'est l'approche retenue par exemple dans ECOMATA [LCB+11] qui a proposé d'utiliser le model-checking pour permettre à un utilisateur d'interroger, grâce à un langage de haut niveau, un modèle complexe représentant un écosystème en Nouvelle-Calédonie [BGK04b].

Une partie de cette thèse repart de cette idée, et l'expérimente dans un domaine différent, et qui s'avère aussi plus complexe, celui de la gestion de pâturage. Le modèle doit pouvoir représenter la dynamique de l'herbe sur la prairie, les activités liées au

pâturage (le déplacement des animaux entre les parcelles, le fauchage des parcelles et la fertilisation des parcelles etc.) ainsi que les stratégies des activités de pâturage permettant de décider quand et quelles activités agricoles effectuer. L'utilisation du modèle doit également pouvoir fournir des informations statistiques et calculer les indicateurs permettant d'évaluer les stratégies utilisées.

Allant plus loin, nous proposons d'étudier en quoi les outils proposés pour les systèmes à événements discrets de type synthèse de contrôleur [RW89] peuvent être utilisés pour identifier des stratégies optimales de gestion des agro-écosystèmes. Ceci fait l'objet d'une étude dans le domaine halieutique en s'appuyant sur le travail existant d'EcoMata, ainsi que d'une étude plus approfondie dans le domaine de la gestion de pâturage intégrée dans le logiciel PaturMata. Un des points importants dans le contexte d'aide à la décision qui est le nôtre est que les stratégies identifiées comme optimales puissent s'interpréter facilement par un utilisateur exploitant agricole ou expert de pâturage.

Contributions

Dans la première partie concernant le projet EcoMata, ma contribution a été d'une part d'améliorer la construction automatique du modèle en automates temporisés à partir de paramètres numériques en la rendant nettement plus efficace, et enfin, d'étudier deux méthodes permettant de dégager les stratégies optimales de gestions de pêche. Une des deux méthodes s'appuie sur la technique de la synthèse de contrôleur alors que l'autre méthode utilise une approche "Générer et tester" pour chercher la meilleure politique de pêche.

Dans la seconde partie, concernant la gestion du pâturage, les deux contributions principales sont de proposer une modélisation hybride et hiérarchique et de proposer des méthodes de recherche de stratégies optimales, s'appuyant sur cette modélisation et donc appliquée à la gestion du pâturage.

- Nous proposons une modélisation hybride en automates temporisés afin de tenir compte d'une part de la dynamique propre de la croissance de l'herbe, qui s'exprime naturellement en termes numériques et d'autre part des actions de pâturage qui s'expriment bien en termes d'événements interagissant sur la croissance de l'herbe dans les parcelles. Ce modèle est ainsi une combinaison du modèle numérique de la prairie et du modèle qualitatif des activités de pâturage. Ce modèle est d'autre part hiérarchique, dégageant le niveau horloge qui gère la croissance de l'herbe, le niveau stratégique ou contrôle, le niveau activités de pâturage et le niveau parcelle. Cette combinaison permet de préserver au maximum la précision de la simulation et la performance de l'algorithme de model-checking afin de répondre aux requêtes de l'utilisateur et analyser les différents scénarios de gestion.
- Nous proposons d'autre part quatre méthodes de recherche de stratégies optimales de la gestion de pâturage, en essayant de profiter au mieux des outils de

type synthèse de contrôleur proposés par le model-checking : une méthode pour la recherche de stratégies optimales de mise au pâturage et trois méthodes de la recherche de stratégies optimales de fertilisation. En particulier, ces trois dernières explorent différentes méthodes informatiques. Une des trois méthodes utilise directement la technique de synthèse de contrôleur pour chercher des stratégies optimales pour une exploitation donnée alors que les deux autres combinent la synthèse de contrôleur et l'apprentissage supervisé pour générer des stratégies génériques par type d'exploitation.

Organisation du manuscrit

La thèse est divisée en quatre parties et sept chapitres. La partie I est composée de deux chapitres. Le chapitre 1 est l'introduction générale de la thèse. Le chapitre 2 sur l'état de l'art présente de manière générale des systèmes d'aide à la décision dans le domaine de l'agro-écologie ainsi que l'utilisation de la technique du model-checking et de synthèse de contrôleur dans un contexte d'aide à la décision

La partie II est consacrée aux travaux effectués autour d'EcoMata [LCB+11], dans le domaine de la gestion de pêche. Il décrit le contexte de mes travaux dans le domaine et se termine par les extensions que j'ai effectuées, en particulier concernant la recherche de stratégies optimales de gestion. Le chapitre 3 présente la modélisation qualitative en automates temporisés d'un réseau trophique de type proie-prédateur soumis aux perturbations environnementales et aux pressions liées aux activités de pêche. Le chapitre 4 présente la transformation automatique du modèle de réseau trophique en automates temporisés de jeux. Un premier algorithme utilise la synthèse de contrôleur. Il permet de générer automatiquement un ensemble de stratégies de gestion de pêche pour que l'état de l'écosystème n'entre pas dans certains états non souhaitables. Il est ensuite comparé à une second algorithme de type "générer et tester".

La partie III présente une modélisation qualitative de la gestion de pâturage en automates temporisés et les méthodes de recherche de stratégies optimales de la gestion de pâturage. Le chapitre 5 présente en détail la modélisation du modèle du pâturage en automates temporisés. Le chapitre 6 présente quatre méthodes de recherche de stratégies optimales dont une méthode est appliquée la mise au pâturage et trois méthodes sont appliquées la fertilisation.

Le manuscrit se termine par une conclusion dans le chapitre 7 de la partie IV.

Chapitre 2

État de l'art : Aide à la décision dans les agro-écosystèmes

2.1 Introduction

Un certain nombre de travaux ont porté sur l'aide à la décision pour la gestion de systèmes complexes, et en particulier la gestion des agro-écosystèmes qui est celle qui nous intéresse dans cette thèse. Un certain nombre de ceux-ci s'appuient directement sur la connaissance du domaine (expertise) comme les systèmes experts qui cherchent à optimiser les décisions dans un contexte de conduite d'exploitations ou de planification d'itinéraires techniques. Nous nous focalisons ici sur ceux qui s'appuient sur un modèle, que ces modèles soient à base d'équations comme c'est le cas de TNT [BDR+02], ou des modèles qualitatifs comme par exemple le modèle Sacadeau [CGG+05], ou des systèmes à événements discrets, DEVS, réseaux de Petri [AMBD98], automates temporisés [AD94], etc. Nous privilégions en particulier les travaux ayant utilisé ces modèles pour aider à la décision et donnons un aperçu des méthodes utilisées. Ceux-ci en général combinent des techniques de stockage de données, d'apprentissage de connaissances, de fouille de données, de visualisation et de recommandation d'actions.

Nous présentons d'abord quelques systèmes d'aide à la décision dans le domaine de la gestion des agro-écosystèmes, puis plus précisément dans le domaine de la gestion du pâturage qui a été le contexte de l'essentiel de ma thèse. Nous terminons en considérant les travaux qui ont utilisé les techniques de model-checking et de synthèse de contrôleur pour aider l'utilisateur à mieux comprendre un domaine et à prendre les décisions en meilleure connaissance de cause.

2.2 Simulation pour l'aide à la décision

La simulation est couramment utilisée dans le domaine de l'agronomie pour aider les chercheurs et les gestionnaires à mieux comprendre les liens complexes entre les actions humaines, le contexte environnemental et les réponses de l'agro-écosystème. Les simulations peuvent être utilisées dans un contexte d'aide à la décision, par exemple pour prédire ce qui va résulter d'actions envisagées ou pour suggérer les actions qui pourraient améliorer la situation.

Nous présentons ci-dessous quelques exemples en insistant sur les techniques utilisées pour transformer un modèle de simulation en un outil d'aide à la décision. Les deux premières approches sont générales au domaine agro-environnemental. Les trois dernières concernent plus particulièrement l'élevage et le pâturage qui est le thème de ma thèse.

2.2.1 Projet Sacadeau

Le projet SACADEAU (Système d'Acquisition de Connaissances pour l'Aide à la Décision sur la qualité de l'EAU) [CGG+05], initié en 2002 lors d'une collabaration entre l'équipe DREAM d'IRISA et l'unité SAS de l'INRA de Rennes, a pour le but de fournir un outil d'aide à la décision pour améliorer la qualité de l'eau contaminée par les pesticides appliqués pour la culture de maïs dans le bassin versant de la région de Frémeur. SACADEAU regroupe trois fonctionnalités principales : la simulation, l'apprentissage et la recommandation d'actions. Un logiciel SACADEAU-SOFTWARE [TMC+12] a été développé regroupant ces trois fonctionnalités et propose une interface graphique qui facilite l'utilisation des trois modules et la visualisation des résultats (cf figure 2.1).

Simulation

À partir d'un modèle numérique d'altitude du bassin versant du Frémeur et d'une carte d'occupation du sol, des arbres de drainage sont construits, qui modélisent le ruissellement. On construit ensuite un ensemble d'arbre d'exutoires de parcelles à partir des arbres de drainage et de la connectivité entre les parcelles [AGOS+09]. On obtient une représentation du bassin versant à l'aide de structures d'arbres où les nœuds sont des exutoires et les arcs représentent la relation "se déverse dans" entre deux exutoires. Un modèle biophysique [GOAC+09] calcule, à l'aide de cette structure, le transfert et la concentration de pesticides dans le flux d'eau et donc la qualité de l'eau arrivant à l'exutoire.

Un modèle décisionnel [SMGOG+11] représente les processus de décisions des agriculteurs concernant la culture du maïs et simule l'application des stratégies de gestion phytosanitaire pour chacune des parcelles cultivées en maïs. Ces stratégies sont décrites sous forme d'*itk* (itinéraire technique) qui est une séquence d'actes techniques effectués par l'agriculteur.

L'utilisateur peut lancer une simulation après avoir spécifié le climat et choisi l'ensemble d'*itk*s sur une interface graphique. À la fin de la simulation, le taux de transfert

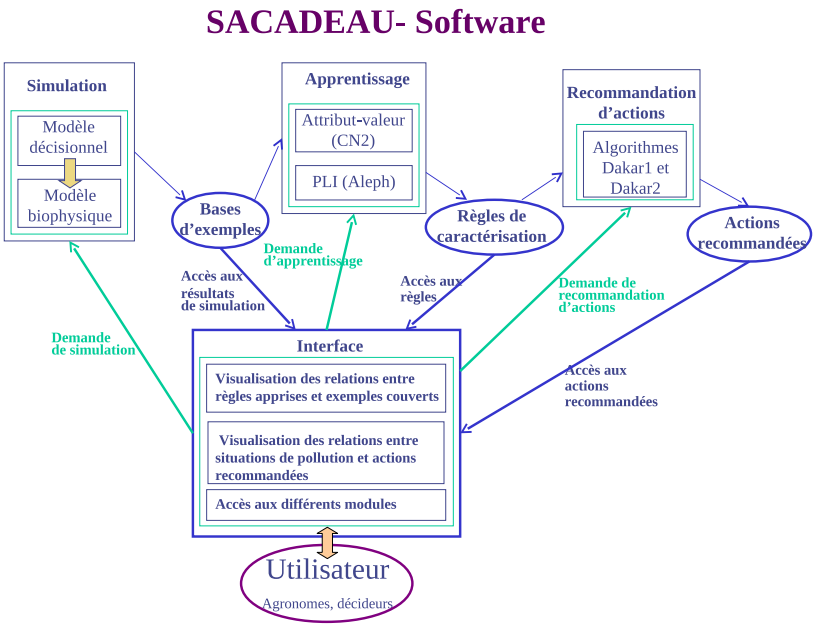


FIG. 2.1 – Architecture du Sacadeau-Software

de pesticide pour chaque arbre d'exutoires est affiché sur une carte géographique. L'utilisateur peut agrandir la carte pour voir les informations détaillées pour chaque exutoire et la partie de parcelle drainée par cet exutoire [TMC+12].

Apprentissage

L'apprentissage sert à mieux comprendre l'impact des différentes stratégies sur le transfert des pesticides. Le but est de caractériser les arbres d'exutoires favorables ou défavorables au transfert des pesticides [Tré08].

La base d'apprentissage contient les exemples provenant des simulations pour le bassin versant du Frémeur. Onze attributs agrégats ont été choisis pour décrire un arbre d'exutoires. Ces attributs concernent les caractéristiques topographiques de l'arbre d'exutoires et les pratiques agricoles sur cet arbre d'exutoires. Par exemple, le pourcentage de matière active utilisée à risque fort, le pourcentage de surface cultivée en maïs etc. Finalement, une étiquette nominative décrivant le taux de transfert (transfert important, transfert faible) sert d'identifiant à la classe. Comme une simulation concerne un ensemble d'arbre d'exutoires, seuls les arbres dont au moins un des exutoires est dans une parcelle cultivée en maïs sont retenus.

L'algorithme d'apprentissage supervisé de règles CN2 [CB91] a été utilisé dans SACADEAU-SOFTWARE. Une fois l'apprentissage terminé, l'utilisateur peut visualiser les règles apprises et les exemples que chaque règle couvre [TMC+12].

Recommandation d'actions

Étant donné une situation proposée par l'utilisateur, comme étant une situation où le taux de transfert de pesticide est important, des actions sont proposées, à partir des règles issues de l'apprentissage, pour améliorer la situation de pollution [TSAC+13]. Les actions proposées s'expriment sous forme de modifications des valeurs d'attributs agrégats utilisés dans la partie apprentissage. Par exemple, une action proposée peut être : mettre un dispositif tampon (bande enherbée ou dispositif équivalent) dont la largeur est comprise entre 16,5 mètres et 67 mètres.

Deux algorithmes de recommandation ont été proposés : Dakar1 et Dakar2. Dakar1 demande d'abord à l'utilisateur de spécifier l'ensemble des attributs que l'utilisateur pourra réellement modifier. Dakar1 cherche ensuite des valeurs pour les attributs modifiables pour que la situation après les changements soit évaluée non polluante. Il est possible qu'aucune solution n'existe pour un ensemble d'attributs modifiables donné. Dakar2 demande à l'utilisateur d'associer un poids à chaque attribut et de saisir un seuil δ. Dakar2 cherche ensuite des valeurs pour un ensemble d'attributs telle que la situation avec les nouvelles valeurs est évaluée non polluante et la somme des poids des attributs modifiés ne dépasse pas δ. Dans certains cas, plusieurs solutions peuvent être proposées s'il en existe.

Sacadeau-Software permet aux utilisateurs de choisir l'algorithme de recommandation, saisir les paramètres de recherche de recommandation et affiche, à la fin de recherche, l'ensemble d'attributs à modifier et leurs nouvelles valeurs.

Le projet Sacadeau s'appuie sur un modèle couplant un modèle biophysique et un modèle décisionnel et permettant de simuler le transfert de pesticides dans un bassin versant breton. Il propose des outils d'aide à la décision, l'apprentissage de règles à partir des résultats de simulations et la recommandation d'actions pour améliorer une situation défavorable.

2.2.2 Moderato

Moderato est un travail issu d'une collaboration entre plusieurs unités de recherche : INRA de Toulouse, ITCF et AGPM de Baziège, INRA ESR de Grignon [BDD+01]. Ce travail s'attaque au problème de la gestion d'irrigation sur les champs de maïs dans la région du sud-ouest de la France où l'apport de pluie est très limité durant les cinq mois d'été (entre mai et septembre). Ce système prend également en compte une large variabilité de climat d'une année à l'autre.

Moderato simule la croissance de culture et les changements d'état du sol sous différentes stratégies de gestion d'irrigation. Le système est composé de cinq composants :

- le contexte hydraulique est une description de l'équipement et de la ressource en eau ;
- le modèle d'action est un modèle décisionnel décrivant les règles de gestion d'irrigation ;
- les modèles biophysiques décrivent les principaux processus liés à l'irrigation et

coucernant le sol et les cultures ;
- l'horloge modélise l'avancement du temps ;
- les fonctions d'évaluation permettent d'analyser les résultats de la simulation du côté économique et également du côté environnemental.

Les utilisateurs décrivent leurs stratégies d'irrigation par cinq règles élémentaires d'irrigation : (1) la règle d'irrigation pour faciliter la semaille, (2) la règle de commencement d'une période d'irrigation, (3) la règle de commencement d'une nouvelle période d'irrigation, (4) la règle d'arrêt d'une période d'irrigation, (5) la règle de délai d'irrigation en fonction du climat.

Voici un exemple de règle (1) :
- *Cond1* : Le nombre de jours depuis la semaille = 15
- *Cond2* : L'apport de la pluie depuis la semaille < 20mm
- *Action* : Irrigation de 20mm

Ces cinq règles sont composées de la même façon : une condition (*Cond1*) temporelle et une condition (*Cond2*) sur l'état du système concernant la ressource d'eau et une action associée. MODERATO évalue les conditions *Cond1* et *Cond2* avant d'appliquer l'action associée.

MODERATO propose une interface graphique qui permet aux utilisateurs de saisir les informations de culture et de composer les règles d'irrigation. MODERATO produit deux sortes de résultats. D'une part, la chronologie de la dynamique de l'état du sol et de la culture. D'autre part les actions de gestion d'irrigation. Un outil supplémentaire a été développé pour calculer les indicateurs de simulation et les afficher sous forme de graphique. Les utilisateurs peuvent modifier les règles et relancer la simulation.

Extension pour aide à la décision

Une extension de MODERATO a été développée afin d'optimiser les règles d'irrigation [BGL04]. Ce logiciel prend en entrée les définitions de culture et de sol, les règles d'irrigation, les intervalles des paramètres numériques dans les règles et une fonction d'évaluation. L'optimisation s'est réalisée à l'aide de l'algorithme d'optimisation numérique PARTITION-2P. Cet algorithme prend aléatoirement des points échantillons dans les domaines des paramètres. En fonction des valeurs des échantillons, il divise les domaines des paramètres en deux régions. L'algorithme choisit une région parmi toutes les régions découpées et répète ce processus jusqu'à ce que le nombre maximum d'itérations soit atteint ou qu'aucune région ne soit plus intéressante que les autres.

Plusieurs méthodes de partitionnement de domaines ont été proposées. Par exemple, le partitionnement égal divise simplement une région en deux partitions égales alors que le partitionnement par pivot divise une région en deux, telles que $\gamma\%$ de bons échantillons se trouvent dans une région alors que $1 - \gamma\%$ de mauvais échantillons se trouvent dans l'autre région. Plusieurs critères de choix de régions ont été proposés. Par exemple, la sélection gloutonne choisit la région qui contient la plus grande valeur maximale alors que la β-sélection prend en compte la valeur moyenne et l'écart type des échantillons.

Des expérimentations d'optimisation ont été faites pour évaluer la performance de

l'algorithme PARTITION-2P avec les différentes méthodes de partitionnement et de choix de région. Toutes les expérimentations ont été faites sur le même type de sol et de culture, huit paramètres à optimiser (l'apport de pluie pour commencer une irrigation, la quantité d'eau pour la première irrigation, etc.) avec deux millions itérations comme la condition d'arrêt. Il a été conclu [BGL04], à partir des résultats de simulation, que la combinaison de la sélection gloutonne et le partitionnement par pivot a la meilleure performance au niveau de la vitesse de convergence vers la valeur optimale.

MODERATO est un outil d'aide à la décision basé sur simulation pour la gestion d'irrigation de la culture de maïs. Les utilisateurs peuvent tester leurs stratégies d'irrigation sur l'interface graphique de MODERATO et visualiser les résultats de simulations. MODERATO a été couplé avec l'algorithme d'optimisation numérique PARTITION-2P pour explorer automatiquement les valeurs des paramètres de stratégie qui maximisent une fonction d'évaluation donnée.

Des travaux en informatique ont été réalisés depuis des années dans le domaine de l'élevage [MMCD13], en particulier l'élevage bovin. Le principal problème traité par ces travaux est l'analyse de la gestion d'une exploitation pour augmenter la production en s'adaptant aux changements environnementaux (le climat par exemple) et/ou économiques (l'augmentation du prix du fourrage).

2.2.3 SEDIVER

SEDIVER [MMCRD11] est une plateforme de simulation à événements discrets pour simuler les activités liées au pâturage de troupeaux bovins sur les prairies sous différentes stratégies de gestion et différents facteurs externes (climat par exemple). L'originalité de ce logiciel est 1) la représentation explicite de la stratégie de gestion et la coordination des activités temporelles et spatiales, 2) la diversité des types d'herbes, d'animaux, de prairies et la difficulté de gestion à cause de cette diversité.

SEDIVER emprunte le framework DIESE [MCR11] (un framework générique pour la simulation des systèmes de la production agricole) pour modéliser les activités de l'élevage. DIESE propose les types d'éléments de base suivants :

- Les entités : Ce sont les structures des informations ou des états qui peuvent être consultées ou modifiées. Les données météorologiques, les troupeaux et la hauteur d'herbe sur les prairies sont représentés à l'aide d'entités.
- Les processus : Ce sont les fonctions qui mettent à jour les entités. La fonction de la croissance d'herbe est représentée par un processus.
- Les événements : Prédéfinis ou créés par les processus, ils déclenchent les processus à une certaine date. Un vêlage de génisse est représenté par un événement prédéfini à une date précise. Un autre événement déclenche le processus de calcul de la dynamique d'herbe.
- Les activités : Elles décrivent la gestion (i.e stratégie) d'exploitation, par exemple le déplacement des animaux, la reproduction des animaux etc.

Les éléments listés sont décrits en un langage spécialement conçu dans le framework DIESE. Ce langage fournit une possibilité d'exprimer des contraintes pour une activité. Ces contraintes permettent au système de vérifier si l'état du système permet d'exécuter cette activité.

SEDIVER acquiert les informations météorologiques, la description de troupeau, la description des prairies et les stratégies de gestion puis simule le processus de pâturage jusqu'à ce que la date de fin de simulation soit arrivée ou qu'il n'y ait plus d'activité à simuler. À la fin de la simulation, le logiciel rédige un rapport de la dynamique du système pendant la période simulée et/ou des indicateurs. Par exemple, la quantité d'herbe fauchée, la digestibilité d'herbe fauchée, la consommation du fourrage conservé, le taux d'utilisation d'herbe et la production du poids vif des animaux.

Deux stratégies de gestion de pâturage à long terme ont été simulées et comparées [MMCRD11]. La première stratégie est celle réellement appliquée par les exploitants . La deuxième prête plus attention à la diversité des plantes et des prairies. Chaque stratégie a été simulée sept fois sur les données historiques de météo pendant sept ans (1998-2004) de la ville d'Ercé dans le département de l'Ariège. Les résultats des simulations ont montré que la seconde stratégie était meilleure que la première en comparant la quantité d'herbe fauchée et la qualité d'herbe pâturée en particulier pendant les années où le climat est défavorable pour la croissance d'herbe.

SEDIVER est un outil d'aide à la décision pour la gestion de l'élevage bovin basé sur la pâturage. Les utilisateurs peuvent simuler l'application des différentes stratégies de gestion puis les comparer. Ce logiciel ne dispose pas d'interface graphique ce qui peut rendre l'utilisation du logiciel et l'interprétation des résultats difficiles.

2.2.4 SEPATOU

Le modèle SEPATOU [CGMC99] a été développé par les chercheurs de l'unité de BIA (Biométrie et Intelligence Artificielle) de l'INRA de Toulouse. SEPATOU est un simulateur à événements discrets de la rotation de pâturage. Il est capable de reproduire la dynamique journalière des prairies et l'application de stratégies de gestion de pâturage de l'agriculteur. Le système est divisé en trois sous-systèmes interagissant entre eux :

- Le système biologique. Ce sous-système traite la dynamique journalière des processus interactifs. Par exemple la croissance de l'herbe, la consommation en herbe et la production laitière des vaches.
- Le système décisionnel. Les actions de gestion sont gérées par le système décisionnel. Ce système est divisé en deux modules dépendants : le système de planification et le système d'exécution des actions. Le système de planification produit un plan d'action et des contraintes associées. Le système d'exécution décide les actions à apporter en suivant le plan et en vérifiant si l'état du système satisfait les contraintes.
- Le système d'information. Le système d'information interprète l'état du système biophysique et fournit des informations décisionnelles. Le système d'information surveille le système biophysique en permanence. Il informe le système décisionnel

par un événement émis lorsque le système biophysique entre dans un état prédéfini par l'utilisateur.

Les stratégies de gestion de pâturage sont décrites à l'aide d'un langage, nommé LNU, exclusivement utilisé dans SEPATOU. Ce langage propose les mots-clés suivants pour décrire les stratégies :

- *PLANNING RULE* définit les planifications (stratégies à long terme) pendant la saison de pâturage. Le mot-clé *TRIGGER* décrit la période ou le moment de l'application de la planification.
- *ACTING RULE* définit une action à appliquer pour un jour fixé.
- *LANDMARK* marque un état du système biophysique. Le système d'information surveille le système biophysique. Un événement est créé et le système décisionnel est informé lorsque le système biophysique entre dans cet état.
- *FUNCTION* interprète le système biophysique. Ces fonctions peuvent être appelées par les règles de stratégie.

Un logiciel, nommé SEPATOU, propose une interface graphique permettant aux utilisateurs de saisir les éléments suivants :

- la configuration du système biophysique : les parcelles, les troupeaux et le stock de fourrage conservé etc ;
- les stratégies de gestion de pâturage décrites dans le langage LNU ;
- le climat historique ou généré ;
- le choix de variables de simulation à retourner à la fin de simulation, par exemple la quantité d'herbe pâturée.

SEPATOU affiche, à la fin de la simulation, les variables de simulation choisies et la chronologie d'actions de pâturage (mise au pâturage, fauchage etc) sous forme de graphique. L'utilisateur peut évaluer, analyser le résultat de simulation, changer les stratégies de pâturage et enfin relancer une nouvelle simulation.

Extension pour aide à la décision

Une extension de SEPATOU [CGMCD01] a été développée pour optimiser les stratégies du pâturage. Il prend en entrée un modèle de pâturage dont les stratégies de pâturage possèdent des paramètres numériques, par exemple le montant de maïs offert aux animaux comme aliment supplémentaire, un vecteur de valeurs de départ des paramètres à optimiser et une fonction d'évaluation de simulation. L'optimisation se réalise à l'aide de l'algorithme d'optimisation stochastique de *Kiefer-Wolfowtz* [KW52]. Cet algorithme cherche, de façon itérative, à partir du vecteur de valeurs de départ, le vecteur de valeurs de paramètres qui maximisent la fonction d'évaluation en analysant le gradient de la fonction d'évaluation. L'itération s'arrête lorsque le gradient est inférieur à un seuil ou lorsque le nombre d'itérations est atteint.

Des exemples d'optimisation sur trois exploitations différentes de pâturage ont été présentées. Chaque exemple concerne neuf paramètres à optimiser. Les optimisations des trois exemples se sont terminés avec succès après deux millions d'itérations en six jours [CGMCD01]. Cette méthode d'optimisation numérique nécessite un paramétrage et un vecteur de départ bien choisis pour que le vecteur converge rapidement vers le

vecteur optimal. En conséquence, il n'existe pas de paramétrage général.

SEPATOU est un outil d'aide à la décision basé sur la simulation pour la gestion de la rotation de pâturage. Les utilisateurs peuvent tester leurs stratégies sur l'interface graphique de SEPATOU et visualiser les résultats de simulations. SEPATOU a été couplé avec l'algorithme d'optimisation numérique *Kiefer-Wolfowtz* pour explorer automatiquement les valeurs des paramètres de stratégie qui maximisent une fonction d'évaluation donnée.

2.2.5 PATUR'IN

PATUR'IN [DPF01] est un logiciel pour la simulation de la gestion de pâturage sur les prairies. Il a été conçu afin d'aider les éleveurs à mieux organiser leurs activités de pâturage. Ce logiciel possède une IHM intuitive et conviviale qui permet aux utilisateurs de simuler des actions de pâturage. PATUR'IN affiche le résultat de la simulation sur l'IHM dès qu'une action est entrée. Les utilisateurs réalisent les simulations par cette méthode pas-à-pas d'applications d'actions. Nous mettons particulièrement en avant ce logiciel puisqu'il est l'inspirateur de la partie "modélisation de pâturage" dans ma thèse.

Dans PATUR'IN, une exploitation de pâturage est représentée par les éléments suivants :

- Un ensemble de parcelles. Chaque parcelle est caractérisée par la surface en hectares (ha), la hauteur d'herbe (cm), la densité d'herbe (kg MS/cm/ha), et le taux de croissance d'herbe (kg Ms/ha/jour). La densité d'herbe décrit la quantité d'herbe par centimètre de hauteur par hectare de surface. Cette valeur n'est pas constante pendant l'année. Elle est décrite par un tableau des valeurs décadaires. Le taux de croissance d'herbe décrit l'augmentation de la quantité d'herbe par jour par hectare. Cette valeur dépend de la saison. Elle est décrite par un tableau des valeurs décadaires.
- Un ensemble de troupeaux. Chaque troupeau est caractérisé par le nombre d'animaux, le type d'animaux (génisses ou adultes), la production laitière potentielle par animal et le poids vif.
- La condition climatique décadaire. Dans le logiciel PATUR'IN, le taux de croissance d'herbe décadaire est calculé sous la condition climatique moyenne. Pour pouvoir simuler les conditions climatiques exceptionnelles, l'utilisateur peut définir un profil de climat en remplissant un tableau de valeurs décadaires. Les valeurs doivent être choisies parmi les cinq valeurs catégoriques prédéfinies : *Très Défavorable*, *Défavorable*, *Normal*, *Favorable* et *Très Favorable*. Chaque valeur catégorique peut représenter différentes situations. Par exemple *Défavorable* peut signifier le froid pendant le printemps ou la sécheresse pendant l'été.

Le logiciel PATUR'IN est capable de simuler les actions de pâturage suivantes :

- La mise au pâturage. C'est l'action où un troupeau est mis sur une parcelle pour une certaine durée pendant laquelle les animaux se nourrissent directement de l'herbe de la parcelle. Du fourrage conservé et/ou du concentré peut être offert aux animaux pendant cette période. Le fourrage conservé peut remplacer une partie d'herbe que les animaux mangent. Ceci est utile au cas où peu d'herbe

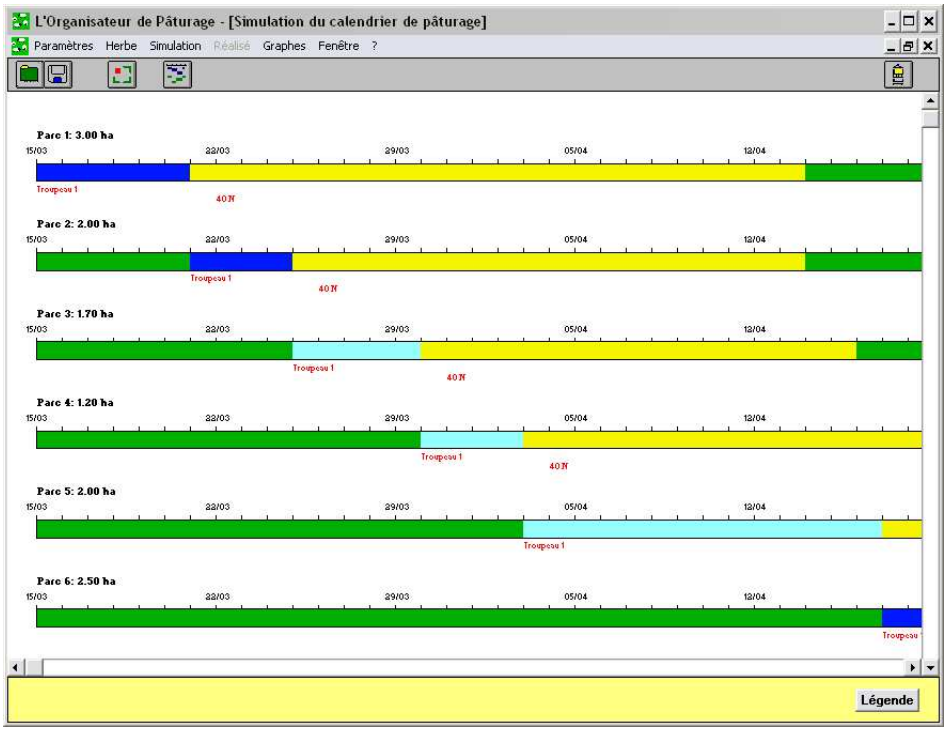


FIG. 2.2 – Exemple d'utilisation du logiciel PATUR'IN. Une simulation d'une exploitation d'un troupeau de 30 vaches laitières adultes sur six parcelles de prairie avec le 15 mars comme date du début de la simulation et pour une durée de 90 jours (partiellement affiché). Les mises au pâturage : du 15/03 au 20/03 sur la parcelle 1 ; du 21/03 au 24/03 sur la parcelle 2 ; du 25/03 au 29/03 sur la parcelle 3 ; du 30/03 et le 02/04 sur la parcelle 4 ; du 03/04 au 16/04 sur la parcelle 5 ; du 17/04 sur la parcelle 6. Les applications d'engrais : 40 kg N/ha respectivement sur la parcelle 1 le 22/03, sur la parcelle 2 le 26/03, sur la parcelle 3 le 31/03 et sur la parcelle 4 le 04/04.

est disponible sur les parcelles. Le concentré sert principalement à stimuler la production laitière et peut aussi remplacer l'herbe.

- Le fauchage. L'action de fauchage simule la coupe de l'herbe. Si l'herbe sur une parcelle dépasse une certaine hauteur, les animaux peuvent rencontrer de la difficulté pour se nourrir. Dans ce cas là, il est préférable de faucher la parcelle et de transformer l'herbe fauchée en fourrage conservé.
- La fertilisation. L'application de l'engrais azoté sur le terrain stimule la croissance de l'herbe. La fertilisation est souvent utilisée où la croissance de l'herbe n'est pas suffisante pour nourrir les animaux. L'augmentation du taux de croissance dépend de la saison et de la quantité d'engrais appliquée sur la parcelle. L'augmentation ne prend effet qu'après une certaine durée à partir de la date de l'application.

PATUR'IN calcule, pour chaque parcelle, la dynamique de l'herbe en prenant la

date, les caractéristiques de la parcelle, la présence des animaux et l'état de fertilisation comme paramètres. Pour une simulation, à partir de la date du début, PATUR'IN calcule jour par jour la hauteur d'herbe jusqu'à la fin de de simulation. Le résultat de la simulation est présenté sur une IHM où l'utilisateur peut consulter et modifier les actions de pâturage. Quand l'utilisateur apporte des modifications, PATUR'IN refait immédiatement le calcul de la dynamique d'herbe et met à jour l'affichage des résultats. L'utilisateur utilise PATUR'IN et simule la gestion de pâturage de manière interactive.

La figure 2.2 présente une simulation dans PATUR'IN d'une exploitation constituée d'un troupeau de 30 vaches laitières adultes et de six parcelles. Chaque parcelle est représentée par une barre colorée horizontale. Différentes couleurs représentent les différents états de parcelle. La couleur verte signifie que la hauteur d'herbe est comprise entre 8 cm et 16 cm. La couleur jaune signifie que la hauteur d'herbe est inférieure à 8 cm. La couleur bleu foncée accompagnée par une étiquette rouge signifie qu'un troupeau dont le nom est indiqué par l'étiquette rouge est mis au pâturage sur cette parcelle. La couleur cyan accompagnée par une étiquette rouge signifie aussi qu'un troupeau est mis au pâturage mais avec du fourrage conservé comme aliment supplémentaire. Les étiquettes rouges "40N" signifient un application de 40 kg N/ha d'engrais azoté sur la parcelle.

Malgré la facilité de l'utilisation du logiciel, PATUR'IN ne propose pas de mécanisme d'expression de stratégies qui permet d'enchaîner automatiquement les actions. Les utilisateurs sont laissés seuls pour explorer eux-mêmes les scénarios. En cas de simulation d'une grande exploitation (une vingtaine de parcelles par exemple), il n'est pas possible d'explorer tous les scénarios envisageables à cause du nombre extrêmement important de possibilités.

PATUR'IN est un logiciel d'aide à la décision pour la gestion de pâturage. Il propose une interface graphique permettant aux utilisateurs de faire les simulations de manière interactive. Ce logiciel ne dispose pas de fonctionnalité d'expression explicite de stratégie contrairement à SEPATOU et SEDIVER. Ceci rend l'évaluation et la comparaison de stratégies très difficiles car les utilisateurs doivent instancier les stratégies par eux-mêmes. PATUR'IN ne propose pas non plus de fonctionnalité d'optimisation de stratégies.

2.3 Model-checking et synthèse de contrôleur pour l'aide à la décision

Il n'est pas très courant de considérer les techniques de model-checking comme outils d'aide à la décision. Cependant quelques travaux ont employé le model-checking pour aider à comprendre voire à contrôler le comportement de systèmes complexes représentés par des systèmes à événements discrets. Nous introduisons le domaine du model-checking et montrons un certain nombre d'applications de ces techniques dans un contexte d'aide à la décision. Le point essentiel est de pouvoir analyser un système complexe en vérifiant des propriétés sur un modèle au lieu de le simuler de manière itérative.

2.3.1 Model-checking pour l'analyse de modèle

Issu du domaine des méthodes formelles, le model-checking [CGP02] est utilisé pour la vérification automatique de systèmes complexes représentés par des systèmes à événements discrets. Lorsque le modèle d'un système est décrit sous la forme d'un automate temporisé, les propriétés sont exprimées à l'aide d'une logique temporelle. Le problème du model-checking peut alors se résumer de la façon suivante : étant donné M un modèle du système et φ une propriété à vérifier, est-ce que M satisfait φ ? Les programmes de model-checking retournent une réponse booléenne (la propriété φ est vérifiée ou ne l'est pas) (cf figure 2.3).

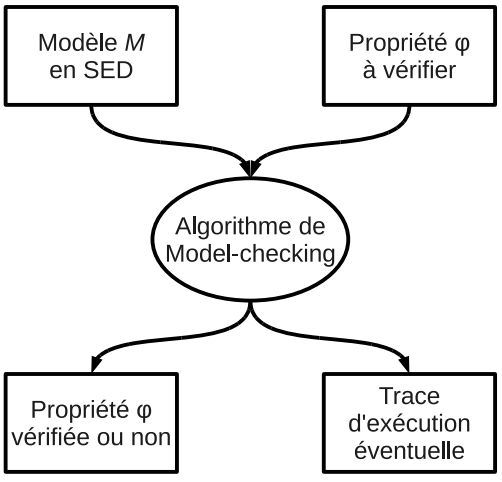


FIG. 2.3 – Algorithme de Model-checking

[CES86] a défini le langage CTL (Computation Tree Logic) dans lequel sont décrites les propriétés φ à vérifier. Les outils les plus connus pour le model-checking symbolique sur les automates temporisés sont KRONOS [Yov97] et UPPAAL [LPY97]. Ce dernier est devenu un logiciel commercial mais reste gratuit pour les utilisations académiques. Il

dispose d'une interface graphique conviviale et facile à utiliser. UPPAAL a été utilisé au long de ma thèse.

S'appuyant sur une modélisation qualitative que l'on peut considérer comme une représentation abstraite d'un système continu, des travaux ont déjà montré l'intérêt d'employer des techniques de model-checking [CGP02, BBF+01] pour vérifier des propriétés du système. La modélisation qualitative peut, en effet, pallier le manque de données nécessaires à la représentation numérique et s'avérer suffisante pour des applications de surveillance ou de diagnostic pour lesquelles la chronologie des événements est essentielle. Nous montrons ci-dessous un certain nombre de travaux où la technique de model-checking est utilisé pour l'aide à la décision.

2.3.1.1 Model-checking dans réseaux d'interaction cellulaire

Des travaux [MRM+08a, MRM+08b] réalisés en 2008 au laboratoire INRIA de Grenoble présentent l'utilisation des systèmes à événements discrets pour la modélisation d'un réseau d'interaction cellulaire et la technique de model-checking pour interroger ce réseau.

Les modèles décrivant la dynamique d'un réseau d'interaction cellulaire sont devenus de plus en plus complexes. La plupart des paramètres sur les gènes, les protéines et les autres molécules sont inconnues malgré une quantité énorme d'information cumulée. Les travaux présentés dans [MRM+08a] portent sur la réponse de la famine de carbone chez la bactérie escherichia coli. Le modèle qualitatif a été choisi comme le plus adapté pour représenter ce système qu'on ne connait qu'imparfaitement. Un programme a été développé permettant de transformer automatiquement un modèle sous forme d'équations différentielles en automates temporisés. Le modèle généré contient approximativement 10^{10} états. La vérification formelle par la machine est donc forcément nécessaire.

Dans ce système, l'aide à la décision est réalisée à l'aide de requêtes en langage naturel permettant d'avoir des informations sur la réponse du réseau cellulaire face à la famine de carbone. Des patterns de requêtes sont choisis à partir des questions les plus fréquemment posées dans le domaine. Un programme interprète la requête en langage CTL et/ou u-calcul. Ceci permet aux utilisateurs non experts en logique temporelle d'interroger le modèle. Par exemple, le pattern "Conséquence" s'exprime : si un état a se produit, sera-t-il possiblement/nécessairement suivi par un état b ? L'utilisateur instancie les états a et b dans ce pattern et formule une requête concrète.

L'originalité de ces travaux consiste à choisir une modélisation de type système à événements discrets pour représenter un système partiellement connu, puis d'interroger ce système à l'aide de requêtes en langage naturel ce qui ne demande pas à l'utilisateur de connaître une logique temporelle.

2.3.1.2 Model-checking pour la simulation de flux de matériel

A. Hélias a proposé, en 2003 dans sa thèse de doctorat, une méthodologie de modélisation qualitative du transfert d'effluents d'élevage sur l'île de la Réunion à l'aide

des automates temporisés [Hél03]. La technique du model-checking a été utilisée pour vérifier les propriétés du système dans un cadre d'aide à la décision pour la gestion de la production et de l'utilisation d'effluents face aux contraintes environnementales et aux règlements associés.

Le transfert d'effluents d'élevage a été modélisé dans un pattern de producteur et consommateur. Les producteurs sont les élevages alors que les consommateurs sont les cultures. Les automates temporisés modulaires sont utilisés pour représenter individuellement chaque producteur et consommateur. Les contraintes temporelles représentent le temps de la production et la fréquence de la consommation.

Le model-checking a été utilisé pour simuler les décisions d'épandages. Les scénarios de gestion sous application des stratégies d'épandage peuvent être simulés à l'aide d'une procédure itérative entre tests et interprétations des résultats à l'aide des connaissances agronomiques. Les expérimentations de simulations sur une période de dix ans, avec KRONOS [Yov97] comme l'outil de model-checking ont été terminées en 47 minutes [HGS08]. L'utilisateur peut ensuite analyser les résultats de simulation et relancer des simulations après avoir modifié les stratégies d'épandage.

Les travaux d' A. Hélias présentent une méthodologie de modélisation qualitative permettant la simulation de la production et de la consommation d'effluents issus d'élevage. L'utilisation de la technique du model-checking permet de simuler les épandages. Un programme itératif sert à tester les stratégies d'épandage.

2.3.2 Synthèse de contrôleur

Ramadge et Wonham [RW87, RW89] ont été les premiers à introduire la notion de contrôlabilité sur les systèmes à événements discrets, avec l'objectif de contrôler le comportement logique d'un système initialement décrit par un langage d'automates. Le système est modélisé par un système à événements discrets qui évolue en générant des événements. L'objectif de cette approche est de synthétiser formellement un contrôleur pour un modèle, tel que le comportement du système supervisé vérifie une propriété donnée (cf figure 2.4). Cette propriété peut décrire des situations à atteindre (atteignabilité) et/ou des situations à éviter (sûreté). Tous les événements émis par le système ne peuvent pas être contrôlés par le contrôleur. Les événements sont donc divisés en deux catégories : événements contrôlables et événements non-contrôlables.

UPPAAL-TIGA est une version dérivée d'UPPAAL spécialement conçu pour effectuer de la synthèse de contrôleur sur un réseau d'automates temporisés [CDF+05]. Même si il est encore en version bêta, il a montré des potentiels dans un certain nombre d'applications que nous présentons dans les sous-sections qui suivent.

2.3.2.1 Synthèse de contrôleur pour l'aide au contrôle de climatisation

Des chercheurs de l'université d'Aalborg au Denmark ont développé une méthode utilisant la synthèse de contrôleur pour réguler la climatisation d'un bâtiment d'élevage [JRLD07].

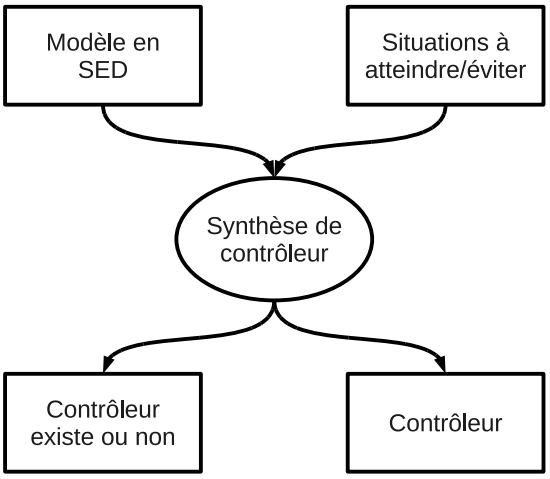


FIG. 2.4 – Algorithme de synthèse de contrôleur

Le contrôle de la température et l'humidité dans un bâtiment d'élevage est important car une température et/ou une humidité trop élevées ou trop basses peuvent conduire à la mortalité des animaux. Pour gérer automatiquement l'environnement d'élevage, la technique de la synthèse de contrôleur a été utilisée pour générer des stratégies de régulation automatique de la climatisation.

Le bâtiment d'élevage est divisé en plusieurs compartiments individuels. Chaque compartiment possède un système de chauffage et des ventilateurs qui permettent d'échanger l'air entre les compartiments voisins. De l'air frais peut être introduit dans un compartiment par évacuer l'air vers un compartiment voisin. Chaque compartiment possède son propre contrôleur de climatisation qui peut allumer/éteindre le chauffage et activer/désactiver les ventilateurs vers les voisins. Par contre, les ventilateurs ne peuvent être activés que si les deux contrôleurs voisins se mettent d'accord. Comme tous les contrôleurs suivent les mêmes stratégies de contrôle, le modèle pour la synthèse contient :

- un automate temporisé modélisant les composants d'un compartiment (ventilateurs et chauffage)
- deux automates temporisés modélisant les états de contrôleurs des compartiments voisins (demande d'activation de ventilateur)
- un automate auxiliaire modélisant le changement environnemental dans le compartiment
- une fonction pour calculer le changement de température

L'outil de synthèse de contrôleur UPPAAL-TIGA a été utilisé pour synthétiser les stratégies de contrôle de chauffage et de ventilation. Les stratégies à la sortie sont transformées dans un format utilisable dans le simulateur SIMULINK. Des simulations faites dans ce simulateur ont prouvé que les stratégies synthétisées par UPPAAL-TIGA peuvent contrôler la température d'un compartiment dans un intervalle souhaité. Malheureuse-

ment les résultats n'ont pas été testés dans la pratique parce que les compartiments ne sont pas parfaitement thermiquement isolés ce qui est une hypothèse de la modélisation.

Ce cas d'étude a montré l'utilisation de la synthèse de contrôleur sur un réseau d'automates temporisés pour l'aide au contrôle de climatisation. Les stratégies générées ont été prouvées, par des simulations, être capables de contrôler correctement le système pour réguler la température et l'humidité.

2.3.2.2 Synthèse de contrôleur pour un système embarqué

Un cas d'étude [CJL+09], réalisée en 2009, présente une utilisation de la synthèse de contrôleur sur un exemple de machine simplifiée composée d'un réservoir d'huile, d'une pompe, d'un accumulateur à huile sous pression et une machine. Le but est de synthétiser un contrôleur pour la pompe afin de maintenir le bon fonctionnement de la machine.

Le système est modélisé en trois automates temporisés. Un automate modélise la machine qui utilise de l'huile suivant une fréquence déterminée et cyclique. Un deuxième automate modélise la pompe qui envoie de l'huile dans un accumulateur d'huile sous pression. L'accumulateur sert de source d'huile pour la machine. Un troisième automate modélise les variables du système, par exemple le niveau d'huile dans l'accumulateur, le compteur du nombre d'activation de la pompe. Parmi ces trois automates, seul l'automate modélisant la pompe est contrôlé par le contrôleur.

Le contrôleur de la pompe doit satisfaire les contraintes suivantes :

- la volume d'huile dans l'accumulateur reste dans un intervalle donné (contraintes de sûreté)
- l'accumulateur doit contenir le moins d'huile possible car plus d'huile se présente dans l'accumulateur, plus de pression se cumule et plus il faut d'énergie pour pomper de l'huile dans l'accumulateur.

La méthodologie développée s'appuie sur le fait que le fonctionnement de la machine est cyclique. Il suffit de synthétiser des stratégies de l'activation de la pompe pour un cycle pour les différents niveaux d'huile présenté dans l'accumulateur au début du cycle. L'ensemble de toutes les stratégies générées forment la stratégie complète pour la pompe.

Uppaal-Tiga a été utilisé pour la synthèse de stratégie dans ce cas d'étude. La stratégie a été comparée avec deux autres stratégies existantes à l'aide du simulateur Simulink. La stratégie générée par Uppaal-Tiga a présenté une meilleure performance en ce qui concerne le montant moyen d'huile dans l'accumulateur.

Ce cas d'étude industrielle a montré l'utilisation de la synthèse de contrôleur pour fabriquer la puce de contrôle d'un système embarqué. Les stratégies synthétisées ont prouvé être capables de contrôler correctement la pompe d'huile et d'être plus performantes en comparaison avec les stratégies existantes.

2.4 Conclusion

Nous avons présenté, dans ce chapitre, des travaux existants portant sur l'aide à la décision pour la gestion des agro-écosystèmes. Le logiciel SACADEAU-SOFTWARE simule, à l'aide d'un modèle biophysique et d'un modèle décisionnel, le transfert et la concentration de pesticide dans le flux d'eau dans la bassin versant de Frémeur. Il peut également recommander des actions pour améliorer la qualité des eaux à l'aide des règles issues de l'apprentissage sur les résultats des simulations. Le logiciel MODERATO simule la croissance de culture et les changements d'état du sol sous différentes stratégies de gestion d'irrigation. Une extension de MODERATO est capable de chercher les valeurs optimales des paramètres des stratégies à l'aide d'un algorithme d'optimisation numérique. SEDIVER est un simulateur à événements discrets pour simuler les activités liées au pâturage de troupeau bovin sur les prairies sous différentes stratégies de gestion. SEPATOU est aussi un simulateur de la gestion de pâturage. Une extension permet d'optimiser les paramètres de stratégies à l'aide d'un algorithme d'optimisation stochastique. PATUR'IN propose une interface graphique intuitive et conviviale qui permet aux utilisateurs d'explorer la gestion de pâturage sur un ensemble de prairies. Malgré la facilité d'utilisation, PATUR'IN ne permet pas d'exprimer les stratégies de gestion de pâturage.

Ces travaux ont pour les plupart utilisé des formalismes de représentation des connaissances ad-hoc pour les partie modélisation et pour la plupart des règles pour l'aide à la décision. A. Hélias propose une modélisation qualitative originale pour le transfert d'effluents d'élevage, en employant les automates temporisés. Cette modélisation permet l'utilisation de la technique du model-checking pour simuler les stratégies de la gestion d'épandage d'effluents. Les travaux réalisés par Pedro T.Monteiro et ses collègues de l'INRIA Grenoble présentent également l'utilisation des automates temporisés pour modéliser les réseaux d'interaction cellulaire et les patterns de requêtes en langage naturel permettant aux utilisateurs d'interroger le système sans avoir à connaître la logique temporelle. Ces travaux ont démontré l'intérêt de ces techniques dans le domaine de l'agronomie et de la biologie. Mon travail se focalise sur l'utilisation de l'automate temporisé dans le domaine de la gestion des agro-écosystèmes, et en particulier la gestion de pâturage.

Nous avons également présenté d'autres travaux qui utilisent la technique de la synthèse de contrôleur basé sur le model-checking. Le premier consiste à utiliser la synthèse de contrôleur pour générer des stratégies pour un contrôleur de la climatisation pour un bâtiment d'élevage. Le second porte plutôt sur la fabrication d'un contrôleur dans un système embarqué pour contrôler une machine industrielle. Il existe peu de travaux basés sur la synthèse de contrôleur car peu d'outils ont été développés pour implémenter cette technique. Dans cette thèse, je propose un approche originale par rapport aux travaux existants. Mon travail étudie non seulement l'utilisation de la technique de la synthèse de contrôleur, mais aussi l'application des stratégies générées dans le cadre de l'aide à la décision dans le domaine de la gestion de pâturage.

Deuxième partie

Modélisation d'un écosystème marin pour gérer la pression de pêche

Cette partie est consacrée à la modélisation d'un écosystème marin dans un contexte d'aide à la décision pour la gestion de la pêche. Nous présentons, dans le chapitre 3, le modèle EcoMata, une modélisation qualitative d'un réseau trophique marin du type proie-prédateur sous pressions environnementales et anthropiques. Ce modèle est exprimé comme un système à événements discrets sous forme d'un ensemble d'automates temporisés. Nous présentons également les scénarios de requête prédéfinis, de type prédictif, exprimés dans un langage de haut niveau permettant aux utilisateurs d'interroger le modèle. Les réponses à ces requêtes sont calculées en utilisant les techniques de model-checking. EcoMata vise à fournir aux gestionnaires de pêche des informations sur la dynamique de la biomasse des poissons sous différents soumis à différentes politiques de pêche. La modélisation en automates temporisés permet d'utiliser un algorithme de model-checking pour vérifier formellement les propriétés du modèle sans faire de simulation. Ma contribution à EcoMata a été le développement d'un prototype logiciel, nommé EcoMata, implémentant cette approche. Ce logiciel permet de générer automatiquement un ensemble d'automates temporisés à partir d'une description du réseau trophique et de lancer des requêtes.

Les scénarios de requêtes prédéfinis dans EcoMata sont de type prédictif "Que se passerait-il si ?" Même si les utilisateurs peuvent tester des politiques de pêche (exprimés sous forme de chronogramme) l'un après l'autre, la recherche des stratégies optimales de pêche reste difficile à cause du nombre important de possibilités. Afin de pouvoir automatiser cette recherche et répondre aux questions du type proactif "Que faire pour", nous avons développé deux méthodes permettant d'identifier le planning optimal. Ces méthodes sont présentées dans le chapitre 4. Une des deux méthodes consiste à utiliser la technique de la synthèse de contrôleur basé sur le model-checking. Cette technique est souvent utilisé pour générer des stratégies de contrôle dans un système industriel et est rarement utilisé comme outil dans un cadre d'aide à la décision. La méthode innovante que nous avons développée utilise cette technique pour chercher des stratégies optimales de pêche pour un réseau trophique modélisé en automates temporisés. Une autre méthode consiste à tester de façon exhaustive tous les plannings de pêche possibles afin d'identifier le meilleur. Cette méthode, plus répétitive, a montré des avantages au niveau de la performance en comparaison avec la méthode utilisant la synthèse de contrôleur.

Chapitre 3

EcoMata - Modélisation d'un écosystème marin en automates temporisés

3.1 Introduction

Nous présentons, dans ce chapitre, un modèle qualitatif d'écosystème de type proie-prédateur, nommé EcoMata, représenté par un ensemble d'automates temporisés. Nous présentons également un ensemble de patrons de requêtes de haut niveau qui permet aux utilisateurs d'explorer l'écosystème. Cette approche, développée par Christine Largouët et Marie-Odile Cordier, a été introduite dans les articles [LCF09, LC10, LCB+11]. Les sections 3.2, 3.3 et 3.4 font un rappel de cette approche. J'ai participé, pendant mon stage de fin d'étude de Master, au développement de l'algorithme de génération d'automates à partir de la définition d'un réseau trophique. Cet algorithme est décrit dans la section 3.5. La section 3.6 présente le logiciel EcoMata qui a constitué une partie de mon travail du stage et le début de ma thèse. La section 3.7 montre les résultats expérimentaux et une analyse des temps de traitement selon la complexité du modèle en automates temporisés. Nous concluons en section 3.8.

3.2 Théorie de l'automate temporisé et du model-checking

Nous présentons, dans cette section, les automates temporisés, formalisme adopté pour la représentation d'un écosystème. Nous présentons ensuite les techniques de model-checking [CGP02] et en particulier les techniques de model-checking symboliques [HNSY94, Yov98] permettant la représentation concise d'un ensemble d'états du système, par opposition à la représentation exhaustive. Le comportement d'un écosystème peut être exploré par la vérification de propriétés décrites en TCTL (Timed Computation Tree Logic) par ces techniques de model-checking symboliques, ce qui évite de devoir balayer l'espace des états de manière systématique. À partir de la syntaxe TCTL, un langage de haut niveau a été conçu. Les utilisateurs peuvent s'appuyer sur ce lan-

gage qui évite l'expression directe des requêtes en logique. Les patrons de requêtes sont présentés dans la dernière partie de cette section.

3.2.1 Automate temporisé

Les automates temporisés [AD94] reprennent le formalisme des automates auxquels s'ajoutent des horloges. Dans le formalisme des automates, un système est représenté par un ensemble d'*états de l'automate* (abrégé par la suite *états*) décrivant les états du système, ces états étant reliés par des transitions étiquetées par des événements. Dans un automate temporisé, la dynamique temporelle est décrite grâce aux horloges qui permettent la définition de contraintes temporelles associées aux états ou aux transitions. Une contrainte temporelle associée à un état est appelée son *invariant*, celle associée à une transition est nommée une *garde*. Il est possible de rester dans un état aussi longtemps que son invariant est vrai. Une transition peut être déclenchée dès que sa garde est vraie. Lors du déclenchement d'une transition, la remise à zéro des horloges est possible.

Définition 3.2.1. On note $\Phi(\mathcal{X}) = \mathcal{X} \times \phi$ où $\phi ::= \{\bigwedge x\ op\ c | x \in \mathcal{X},\ c \in \mathbb{N},\ op \in \{<, \leq, =, \geq, >\}\}$ l'ensemble des contraintes d'horloges. Un automate temporisé est un n-uplet $TA =< \mathcal{Q}, Act, \mathcal{X}, E, \mathcal{I} >$ où :

- $\mathcal{Q}$ est un ensemble fini d'états, $q_0 \in \mathcal{Q}$ étant l'état initial ;
- Act est un ensemble fini d'étiquettes ;
- $\mathcal{X}$ est un ensemble fini d'horloges ;
- $\mathcal{I} \subseteq \mathcal{Q} \times \Phi(\mathcal{X})$ associe à chaque état une contrainte temporelle appelée l'*invariant* de l'état ;
- $E \subseteq \mathcal{Q} \times Act \times 2^{\mathcal{X}} \times \Phi(\mathcal{X}) \times \mathcal{Q}$ est un ensemble fini de transitions. Un tuple $e = \langle q, \sigma, \lambda, \phi, q' \rangle$ représente une transition reliant l'état q vers l'état q' étiquetée par $\sigma \in \Sigma$. ϕ est une contrainte temporelle sur les horloges $\mathcal{X}$ représentant la condition de déclenchement de la transition. λ est l'ensemble d'horloges à remettre à zéro.

Une exécution d'automate temporisé commence toujours par son état initial q_0. Pendant une exécution, une *configuration de l'automate* est dénotée par $\langle q,\ v \rangle$ où q indique un état de l'automate et v représente une valuation des horloges.

Pendant l'exécution d'un automate temporisé, deux types d'évolution sont possibles :

- Passage de temps : Un passage de temps représente une durée de temps $\Delta\tau_i = \tau_{i+1} - \tau_i$ devant s'écouler entre deux configurations $\langle q_i,\ v_i \rangle$ et $\langle q_{i+1},\ v_{i+1} \rangle$ sans violer les invariants de l'état q_i. Formellement, $\langle q_i,\ v_i \rangle \xrightarrow{\Delta\tau_i} \langle q_{i+1},\ v_{i+1} \rangle \Leftrightarrow (q_i = q_{i+1}) \wedge (v_{i+1} = v_i + \Delta\tau_i,\ \forall v, v \in (0,\ \Delta\tau_i] \mid (v_i + v) \models I(q_i))$
- Action discrète : Une action discrète est un déclenchement de transition $\langle q_i,\ v_i \rangle \xrightarrow{e} \langle q_{i+1},\ v_{i+1} \rangle$ si il existe une transition $e = (q_i, \sigma, \lambda, \phi, q_{i+1})$ telle que $(v_i \models \phi) \wedge (v_{i+1} = v_i) \wedge (v_{i+1} \models I(q_{i+1}))$

Une exécution d'un automate temporisé TA est une séquence de telles évolutions qu'elles soient dues au passage du temps ou à des actions discrètes. Formellement, elle se

représente par une suite de configurations partant d'une configuration initiale $\langle q_0,\ v_0\rangle$.

$$Exe(TA) = \langle q_0,\ v_0\rangle \rightsquigarrow \langle q_1,\ v_1\rangle \cdots \langle q_i,\ v_i\rangle \cdots \langle q_n,\ v_n\rangle$$

3.2.2 Réseau d'automates temporisés

La définition d'automates temporisés permet de construire un *réseau d'automates temporisé* par le produit d'un ensemble d'automates temporisés. Le réseau de deux automates $\mathcal{A}_1 =< \mathcal{Q}_1, Act_1, \mathcal{X}_1, E_1, \mathcal{I}_1 >$ et $\mathcal{A}_2 =< \mathcal{Q}_2, Act_2, \mathcal{X}_2, E_2, \mathcal{I}_2 >$ est le produit des deux automates, dénoté par $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$ qui est un automate temporisé $< \mathcal{Q}_1 \times \mathcal{Q}_2, Act_1 \cup Act_2, \mathcal{X}_1 \cup \mathcal{X}_2, E, \mathcal{I} >$ où :

- $\forall\,(q_1, q_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2,\ \mathcal{I}\,(q_1, q_2) = \mathcal{I}_1\,(q_1) \wedge \mathcal{I}_2\,(q_2)$
- E est un ensemble de transitions construit de la façon suivante :
  - pour tout $\sigma \in \Sigma_1 \cap \Sigma_2$, pour tout $\langle q_1, \sigma, \lambda_1, \phi_1, q_1'\rangle \in E_1$ et $\langle q_2, \sigma, \lambda_2, \phi_2, q_2'\rangle \in E_2$, $\langle (q_1, q_2), \sigma, \lambda_1 \cup \lambda_2, \phi_1 \wedge \phi_2, (q_1', q_2')\rangle \in E$
  - pour tout $\sigma \in \Sigma_1 \setminus \Sigma_2$, pour tout $\langle q_1, \sigma, \lambda_1, \phi_1, q_1'\rangle \in E_1$ et tout $q_2 \in \mathcal{Q}_2$, $\langle (q_1, q_2), \sigma, \lambda_1, \phi_1, (q_1', q_2)\rangle \in E$
  - pour tout $\sigma \in \Sigma_2 \setminus \Sigma_1$, pour tout $\langle q_2, \sigma, \lambda_2, \phi_2, q_2'\rangle \in E_2$ et tout $q_1 \in \mathcal{Q}_1$, $\langle (q_1, q_2), \sigma, \lambda_2, \phi_2, (q_1, q_2')\rangle \in E$

Le réseau de n automates temporisés est construit de même façon. Une *configuration de réseau d'automate* est dénotée par $\langle \bar{q},\ v\rangle$ où $\bar{q}$ indique le vecteur d'états des automates et v représente une valuation des horloges.

3.2.3 Model-checking sur TCTL

La logique la plus répandue pour la vérification d'automates temporisés est la logique TCTL (*Timed Computation Tree Logic*) qui est une extension de la logique CTL autorisant l'expression de contraintes temporelles dans les propriétés. Les formules TCTL sont définies de manière inductive par la grammaire suivante définie dans [HNSY94] :

$$f ::= p \mid x \in I \mid \neg f \mid f_1 \vee f_2 \mid \exists\Diamond_I f \mid \forall\Diamond_I f \mid \exists\Box_I f \mid \forall\Box_I f$$

avec $p \in P$ un prédicat de base, $x \in \mathcal{X}$ une horloge et I un intervalle de temps. Intuitivement l'opérateur $\exists\Diamond_I f$ signifie qu'il existe une exécution menant à une configuration satisfaisant la propriété f au temps $t \in I$. $\forall\Diamond_I f$ signifie que chaque exécution possède une configuration où la propriété f est valide au temps $t \in I$. $\forall\Box_I f$ signifie que toutes les configurations sur toutes les exécutions satisfont la propriété f. $\exists\Box_I f$ signifie qu'il existe une exécution telle que tous les configurations de l'exécution satisfont la propriété f.

Un automate temporisé ayant la particularité de posséder un nombre infini de comportements liés à la valuation de ses horloges, on se trouve confronté au problème de l'explosion combinatoire. Afin de résoudre ce problème, le model-checking symbolique représente des ensembles de configuration à l'aide de structures de données efficaces comme les diagrammes de décision binaire (encore appelés BDD pour **Binary Decision Diagram**), ou les matrices de bornes (DBM pour **Difference Bound Matrices**).

Les algorithmes de model-checking symboliques travaillent sur ces ensembles de configurations et déterminent, en fonction des propriétés à vérifier, soit l'espace atteignable, soit l'ensemble des configurations satisfaisant une formule TCTL [BBF+01].

3.2.4 Automate temporisé dans UPPAAL

Uppaal [LPY97] étend le formalisme des automates temporisés présenté précédemment. Nous présentons ici les particularités d'Uppaal.

- État urgent. Les états urgents ajoutent une propriété d'urgence de départ d'un état. Dans une configuration de réseau d'automate temporisé $\langle \bar{q},\ v \rangle$ pendant une exécution, si un ou plusieurs automates temporisés sont dans un état urgent, les seules évolutions partant de cette configuration doivent être des actions discrètes (non nécessairement celles permettant aux automates de quitter les états urgents) jusqu'à ce qu'aucun automate ne soit plus dans un état urgent.
- État engagé. Les états engagés peuvent exprimer l'atomicité de deux transitions. Dans une configuration de réseau d'automate temporisé, si un automate est dans un état engagé, seules les évolutions de type action discrète permettant le départ de l'état engagé sont autorisées. Cette caractéristique permet d'assurer qu'aucun délai, ni aucune autre action discrète n'a lieu entre deux transitions.
- Canaux de synchronisation. Cette caractéristique offre un mécanisme de synchronisation de déclenchement de transition entre deux ou plusieurs automates. Les notations $a!$ et $a?$ représentent respectivement l'événement émis et l'événement reçu permettant une synchronisation sur le canal a. Dans le système global, produit synchronisé de tous les automates, à chaque événement reçu doit correspondre un événement émis alors que les transitions non synchronisées correspondent à une évolution asynchrone du sous-système.
- Variables et tableaux. Des variables peuvent être déclarées dans le système d'automates. Leurs types sont booléen, entier, entier borné ou tableaux de types listés. Par exemple *bool t*; définit une variable booléenne t alors que *int b*[10]; définit un tableau d'entiers de la taille de 10. Les variables peuvent être utilisées pour définir les invariants et les gardes en complément des horloges.
- Paramètres. Il est possible de déclarer des automates avec des paramètres. Les automates initiés avec les différentes valeurs de paramètres montrent différents comportements. Par exemple, dans l'automate présenté dans la figure 3.1, si on remplace la valeur 2 par un paramètre *delai* dans les invariants ($timer <= 2$) et les gardes ($timer == 2$), les automates initiés avec différentes valeurs fonctionnent avec différentes délais.
- Fonctions de calcul. Quand une transition est déclenchée, les instructions de mise à jour des horloges et/ou des variables associées à cette transition sont exécutées. Cependant, ces instructions sont limitées aux instructions d'affectation comme $b[1] = 6$. Afin d'effectuer des calculs plus complexes, on peut déclarer des fonctions dans un langage de programmation proposé par Uppaal qui ressemble au langage C. Par exemple la fonction retourne le *pgcd* entre deux valeurs entières positives :
  $int\ pgcd(int\ a,\ int\ b)\ \{$

$$while(a! = b)\{if(a > b)\ a = a - b;\ else\ b = b - a;\}$$
$$return\ a;\}$$

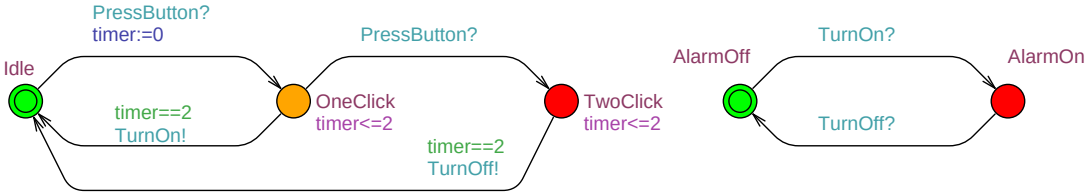


FIG. 3.1 – Deux automates synchronisés. L'automate de gauche convertit un clic et deux clics sur un bouton en deux messages de synchronisation *TurnOn* et *TurnOff*. L'automate de droite reçoit ces messages pour déclencher et couper l'alarme.

3.3 Modélisation d'un écosystème marin en automates temporisés

Nous présentons l'application du formalisme des automates temporisés à la modélisation d'un écosystème. L'illustration est faite sur un écosystème marin simplifié décrit dans la section 3.3.1.

3.3.1 Réseau trophique de l'écosystème

Nous avons appliqué cette approche au domaine de halieutique. Un écosystème récifo-lagonaire de l'atoll d'Uvéa en Nouvelle-Calédonie a motivé et a été modélisé selon cette approche [LCB+11]. Des connaissances issues de précédentes études ont permis de représenter cet écosystème de manière qualitative (cinq groupes trophiques et trois forces de pêches) puis d'étudier l'impact d'un accroissement de l'effort de la pêche (l'écosystème n'étant aujourd'hui touché que par une pêche de subsistance).

Le modèle halieutique utilisé a été volontairement simplifié (cf figure 3.2). Dans cet exemple, l'écosystème est représenté par un réseau trophique composé de quatre espèces de poisson (le thon, le maquereau, la sardine et l'anchois), deux pressions de pêche (sur le thon et le maquereau) et une perturbation environnementale (le réchauffement climatique). Les espèces sont considérées comme des compartiments trophiques qui échangent des flux de biomasse *via* un processus de prédation. Le flux de biomasse entre la proie et le prédateur est représenté par la flèche reliant les deux espèces. Par exemple, le maquereau se nourrit de sardine et d'anchois alors que le thon est le prédateur du maquereau. La pêche thon et la pêche maquereau prélèvent respectivement une partie de la biomasse du thon et du maquereau. Une flèche pointillée reliant le réchauffement et l'anchois indique que cette perturbation s'applique sur l'espèce d'anchois qui cause une mortalité sans transformation de biomasse.

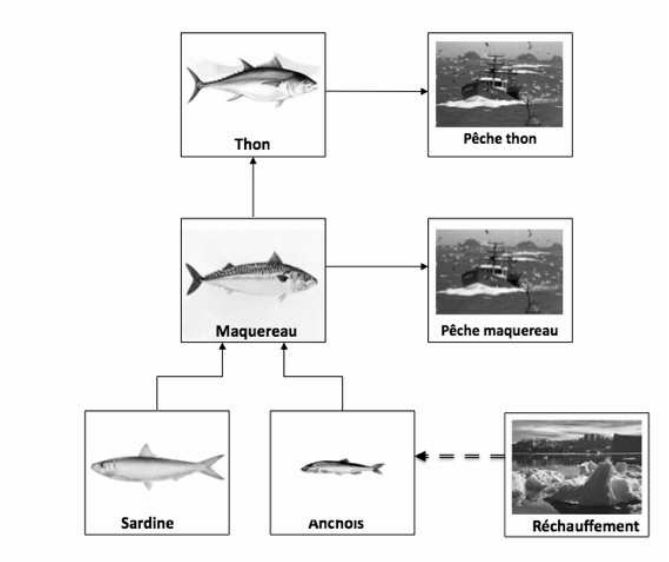


FIG. 3.2 – Système halieutique simplifié : l'écosystème thon-maquereau

3.3.2 Modélisation de l'écosystème

La modélisation proposée s'appuie sur une approche générique [LC10] dans laquelle l'écosystème est représenté de manière modulaire comme un ensemble d'espèces interagissant avec lesquelles se synchronisent d'autres modules représentant les perturbations anthropiques et naturelles. L'intérêt de l'approche est de pouvoir représenter toutes les entités de l'écosystème dans un formalisme unifié permettant l'expression des contraintes temporelles propres à chaque sous-système. La modélisation est qualitative et permet de représenter à la fois le réseau trophique et les modules externes. Un réseau trophique de type proie-prédateur se prête bien à la modélisation qualitative par systèmes à événements discrets grâce à : 1) la discrétisation possible de ses valeurs de biomasse en niveaux de type *Low*, *Normal* ou *High* et 2) la possibilité de représenter la dynamique liée à l'interaction proie-prédateur par des événements agissant sur l'évolution des stocks.

Le comportement du réseau trophique de la figure 3.2 est représenté par quatre automates, chacun étant associé à une espèce du réseau trophique. Pour chacune des espèces, l'automate décrit l'évolution de la biomasse entre les trois états qualitatifs (*Low*, *Normal* et *High*). L'évolution entre ces états qualitatifs est graduelle (impossibilité de passer de l'état *Low* à l'état *High* directement par exemple). Les états directement accessibles depuis un état qualitatif s'appellent les états *voisins*. Le changement d'état qualitatif d'une espèce est déclenché par un événement de synchronisation émis par un automate en interaction avec cette espèce (proie, prédateur, pression de pêche et perturbation environnementale). Le passage entre deux états qualitatifs de biomasse n'est pas immédiat. Il est progressif avec un délai dépendant de l'espèce et du type de pres-

sion. On utilise les états de l'automate (en blanc sur la figure) pour représenter les *états stables*, les états qualitatifs de la biomasse qui n'évoluent pas avec le temps et les états de l'automate (en noir) pour représenter les *états intermédiaires*, les états modélisant une évolution entre deux *états stables*. Le délai de l'évolution est représenté, en tenant compte de l'incertitude liée à la connaissance des espèces, par la garde partant de l'état intermédiaire et par l'invariant qui figure sur ce même état. Une fois le délai écoulé, le système atteint un nouvel état qualitatif de biomasse. Il déclenche un événement de synchronisation indiquant son entrée dans ce nouvel état qualitatif de biomasse. Cet événement peut déclencher à son tour des évolutions dans les autres automates.

La modélisation permet de représenter des annulations ou des changements de durée d'évolution (en cas de pression double, par exemple, comme l'augmentation de l'état qualitatif d'un prédateur se cumule dans un deuxième temps à une forte pression de pêche). Des transitions entre états intermédiaires doivent donc être possibles, ainsi que des retours vers les états stables d'origine. Les retours se font sans délai.

L'application du formalisme est illustrée pour l'espèce *thon* subissant la pression *pêche_ thon* (cf. figure 3.3). L'état initial, représenté par un double cercle est l'état qualitatif *Normal*. Lorsque la pression de pêche diminue (événement *peche_ thon_ low ?*), le système rentre dans un état intermédiaire dont la contrainte temporelle est *t<=40*. La transition partant de cet état intermédiaire et arrivant à l'état *thon_ High* est étiquetée par la garde *t>=36*. Ces deux contraintes signifient que la biomasse du thon peut atteindre l'état qualitatif *thon_ High* au plus tôt en 36 unités de temps et au plus tard en 40 unités de temps. Le choix du moment du passage de l'état qualitatif de biomasse est non déterministe. Cette caractéristique permet de simuler l'incertitude dans un modèle d'écosystème. Lorsque le thon atteint l'état qualitatif la biomasse *High*, il génère un événement *thon_ high !* qui peut modifier par synchronisation l'évolution de sa proie, le maquereau, dont le modèle n'est pas donné ici. L'annulation de cette diminution de la pression de pêche (représentée par l'événement *peche_ thon_ normal ?*) provoque le retour de l'espèce dans son état qualitatif initial (*Normal*). Au contraire, si la pression de pêche est modifiée et passe de faible à nulle (événement *peche_ thon_ stop ?*), le passage vers l'état qualitatif *High* est accéléré (transition entre états intermédiaires).

Les forces de pêches comme les pressions naturelles sont également représentées par des automates décrivant des niveaux qualitatifs et des contraintes de temps décrivant la durée de la pression ou le délai d'activation du réchauffement climatique.

3.4 Patrons de requêtes en TCTL

Des patrons de requêtes permettant l'interrogation d'un écosystème ont été proposés dans [LC10, LCF09]. Ces requêtes constituent un langage de haut-niveau plus facile à appréhender par les utilisateurs. Les questionnements, correspondant à la simulation de scénarios prédictifs, sont de type : "étant donné une situation initiale et une politique appliquée, que va-t-il se passer si ?". On appelle *situation S* l'état global de l'écosystème représenté par une valeur qualitative de la biomasse pour chacune des espèces (la situation pouvant définir l'état d'une seule espèce jusqu'aux états de toutes

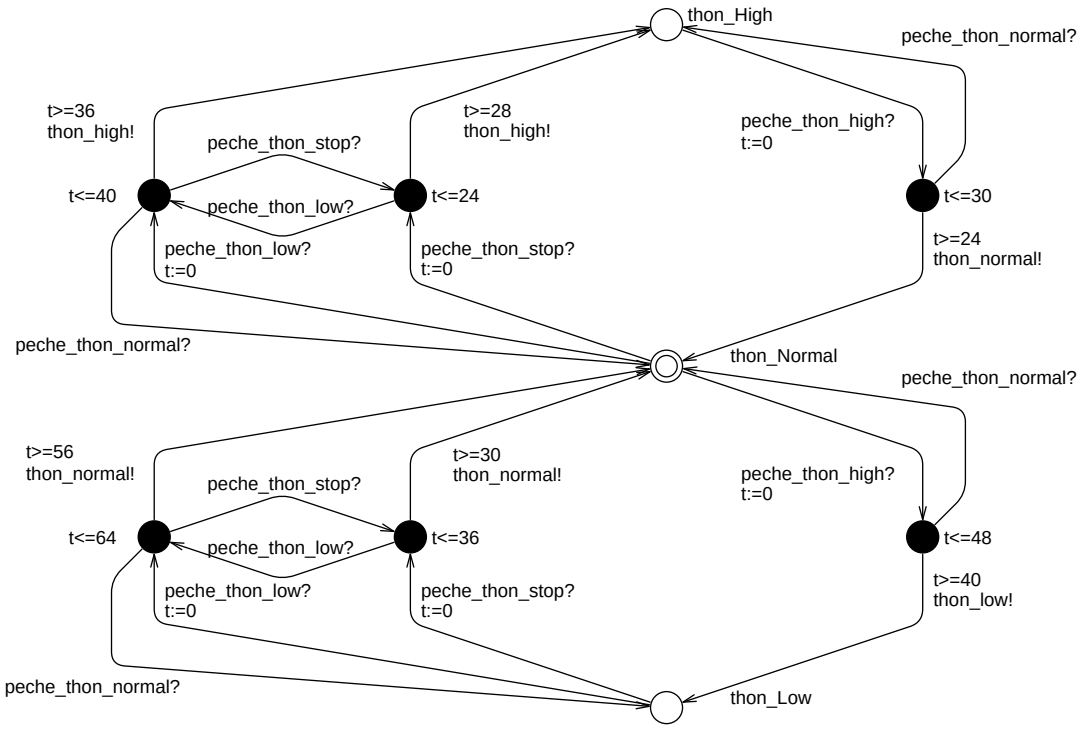


FIG. 3.3 – Exemple d'automate modélisant l'espèce thon

les espèces). Une situation correspond à un ensemble d'états de réseau d'automates temporisés. Afin de bénéficier de toute l'efficacité des techniques de model-checking permettant l'exploration de systèmes temporels à taille réelle, ces patrons de requêtes ont été exprimés à l'aide de la logique TCTL et sont décrits dans le tableau 3.1. Grâce à UPPAAL, on peut exprimer une situation dans une formule TCTL par les valeurs qualitatives de la biomasse pour chacune des espèces sans avoir à énumérer tous les états de réseau d'automates temporisés correspondant à cette situation. Pour chaque patron, on définit l'objectif de la requête, ses données en entrée, la formulation de la requête en logique TCTL et les résultats produits. Les patrons *Reachability*, *Safety* et *Always* prennent en entrée une situation et sont traduits en une seule propriété TCTL vérifiée par model-checking : la réponse retournée est donc classiquement *vrai* ou *faux*. Les patrons *WhichStates* et *WhichDate* prennent en entrée respectivement une date t et une situation et sont implémentés par une itération de formules TCTL. Une information plus précise en réponse à la requête peut alors être retournée, c'est pourquoi les patrons *WhichStates* et *WhichDate* renvoient respectivement les situations possibles et une date. Des exemples d'utilisation de ces patrons sont présentés dans la section 3.6.

Patron	Description	Formulation TCTL dans UPPAAL	Résultats
Atteignabilité (**Reachability**)	Teste si une situation S est possible à une date t (éventuellement non précisée)	$\exists\Diamond(S \wedge chrono.t = t)$ $\exists\Diamond(S)$	vrai ou faux
QuelsEtats (**WhichStates**)	Recherche toutes les situations S_i à une date t	Propriété de type atteignabilité itération de la formule sur les S_i $\exists\Diamond(S_i \wedge chrono.t = t)$	Les situations S_i au temps t
QuelleDate (**WhichDate**)	Recherche la date t_i de la première occurrence d'une situation S (depuis S_{init})	Propriété de type atteignabilité itération de la formule sur les t_i $\exists\Diamond(S \wedge chrono.t = t_i)$	Si S est atteignable, retourne le premier t_i
Sûreté (**Never**)	Vérifie si une situation indésirable S ne peut pas se produire	$\forall\Box(not\ S)$	vrai si S n'arrive jamais, faux sinon
Toujours (**Always**)	Vérifie qu'une situation S est toujours vérifiée	$\forall\Box(S)$	vrai si S est toujours satisfaite, faux sinon

TAB. 3.1 – Les cinq patrons de scénario définis pour explorer l'écosystème

3.5 Génération automatique d'un réseau d'automates

Comme décrit dans la section 3.3, les états qualitatifs d'espèce sont représentés par les états de l'automate. La garde sur les transitions et les invariants sur les états permettent d'exprimer les contraintes temporelles associées à l'évolution de la biomasse de l'espèce. Ces contraintes temporelles sont calculées à l'aide du modèle numérique *Lotka-Voltera*. Afin de rendre plus facile l'élaboration du réseau d'automates temporisés et dans le cas d'un écosystème pour lequel on dispose d'un modèle numérique, un algorithme générique permettant de générer ce réseau à partir de la définition d'un réseau trophique a été développé. Par contre, cet algorithme de génération initialement développé avant mon stage n'était pas suffisamment efficace au niveau du temps d'exécution pour une utilisation interactive. Une de mes contributions a été d'améliorer la performance du générateur. Le résultat a été satisfaisant : le temps d'exécution de la génération d'un ensemble d'automates est passé de quelques heures à moins de 30 secondes. Nous présentons, dans cette section, l'algorithme de génération à l'issu de mon travail de stage.

3.5.1 Modèle numérique Lotka-Voltera

Les équations de Lotka-Volterra, que l'on désigne aussi sous le terme de "modèle proie-prédateur", sont un ensemble d'équations différentielles non linéaires du premier ordre. Elles sont couramment utilisées pour décrire la dynamique de systèmes biolo-

giques dans lesquels un prédateur et sa proie interagissent [Mur02].

$$\begin{cases} N(0) = N_0,\ P(0) = P_0, \\ \dfrac{\mathrm{d}N(t)}{\mathrm{d}t} = N(t)(\alpha - \beta P(t)) \\ \dfrac{\mathrm{d}P(t)}{\mathrm{d}t} = P(t)(\delta N(t) - \gamma) \end{cases}$$

où :

- $N(t)$ est l'effectif des proies, $P(t)$ l'effectif des prédateurs, N_0 la population initiale des proies, et P_0 la population initiale des prédateurs ;
- t est le temps, $\dfrac{\mathrm{d}N(t)}{\mathrm{d}t}$ et $\dfrac{\mathrm{d}P(t)}{\mathrm{d}t}$ représentent la croissance des populations au cours de temps,
- α est le taux de reproduction des proies en absence de prédateur, β le taux de mortalité des proies due aux prédateurs, δ est le taux de reproduction des prédateurs en fonction des proies mangées, et γ le taux de mortalité des prédateurs en l'absence de proies.

Pour une espèce i, l'évolution de la biomasse $B_i(t)$ dans le temps peut être exprimée à l'aide de l'équation différentielle suivante :

$$\frac{\mathrm{d}B_i(t)}{\mathrm{d}t} = B_i(t)(\sum_{k}^{h} \delta_{ik} B_k(t) - \sum_{k}^{j} \beta_{ik} B_k(t) + M_0) \tag{3.1}$$

où :

- h est l'ensemble des proies de l'espèce i et j est l'ensemble des prédateurs ;
- $B_k(t)$ est la biomasse de l'espèce k au temps t ;
- δ_{ik} est la croissance de la biomasse de l'espèce i à partir de la prédation de l'espèce k ;
- β_{ik} est la mortalité de l'espèce i à cause de la prédation par l'espèce k ;
- M_0 est une constante qui représente la croissance et la mortalité dues aux proies et prédateurs qui ne sont pas présents dans le réseau trophique et qui sont considérés comme constants pendant la simulation.

Une caractéristique de la modélisation qualitative est que la biomasse d'une espèce de poisson est discrétisée en niveaux qualitatifs. Une hypothèse a été introduite qui considère que la biomasse des espèces voisines reste constante tant que ces espèces ne changent pas d'état qualitatif. L'équation 3.1 devient :

$$\frac{\mathrm{d}B_i(t)}{\mathrm{d}t} = B_i(t)(\sum_{k}^{h} \delta_{ik} B_k(t_0) - \sum_{k}^{j} \beta_{ik} B_k(t_0) + M_0) \tag{3.2}$$

où t_0 est le moment du début de l'évolution de biomasse de l'espèce i. Étant donné que la biomasse de l'espèce i est B_0 au temps t_0, la fonction de l'évolution de la biomasse de l'espèce i est :

$$B_i(t) = B_0 e^{(\sum_k^h \delta_{ik} B_k(t_0) - \sum_k^j \beta_{ik} B_k(t_0) + M_0)(t - t_0)} (t >= t_0) \tag{3.3}$$

Posons $B_i(t_1) = B_1$ où B_1 est la biomasse de l'espèce i d'un état qualitatif voisin, la durée de l'évolution t_d est :

$$t_d = t_1 - t_0 = \frac{lnB_1 - lnB_0}{\sum_k^h \delta_{ik} B_k(t_0) - \sum_k^j \beta_{ik} B_k(t_0) + M_0} \tag{3.4}$$

L'avantage le plus important de l'hypothèse introduite est que l'équation peut être résolue de manière formelle comme décrite ci-dessus, ce qui évite une résolution numérique coûteuse en temps, et raccourcit de manière remarquable le temps de génération des automates.

La formule 3.4 est conçu pour calculer le temps d'évolution de la biomasse dans une situation donnée (l'état qualitatif des prédateurs, des proies et des perturbations). Or il est possible que la situation change pendant l'évolution. Nous présentons dans la section 3.5.2 la solution modélisant ce changement d'évolution.

Un intervalle de confiance a été ajouté aux valeurs de biomasse B pour chaque espèce et aux valeurs de δ et de β. Cet intervalle peut être personnalisé pour simuler différentes incertitudes.

$$t_d = t_1 - t_0 = \frac{ln(\sigma_i B_1) - ln(\sigma_i B_0)}{\sigma_{\delta i} \sum_k^h \delta_{ik} B_k(t_0) - \sigma_{\beta i} \sum_k^j \beta_{ik} B_k(t_0) + M_0} \tag{3.5}$$

En utilisant la bonne valuation des σ, une valeur maximum t_{dmax} et une valeur minimum t_{dmin} peuvent être calculées à partir de l'équation 3.5. Comme expliqué en section 3.3.2, les évolutions de la biomasse sont représentées par le passage d'un état qualitatif dit stable à un état qualitatif voisin stable, au travers d'un état dit intermédiaire. La contrainte temporelle associée à cet état intermédiaire (invariant) et celle associée à la transition reliant l'état intermédiaire et l'état qualitatif voisin (garde) décrivent l'intervalle de la durée de l'évolution. La valeur t_{dmin} est associée à la garde de la transition et la valeur t_{dmax} est associée à l'invariant de état intermédiaire.

3.5.2 Algorithme

Nous illustrons le processus de génération d'automates. Il commence par la construction proprement dite des automates, successivement pour chacune des espèces. Cela inclut la création des états, stables et intermédiaires, et des transitions entre ces états. Cette étape utilise les équations de Lotka-Volterra pour identifier les changements et les contraintes temporelles associées comme expliqué en 3.5.1.

La génération automatique conduit en général à un ensemble d'automates dont le nombre d'états et de transitions est important. Afin de diminuer la complexité du modèle, et d'améliorer en conséquence le temps de réponse aux requêtes, il est possible ensuite de simplifier le modèle en regroupant les états stables similaires, pour réduire le nombre d'états et de transitions des automates obtenus. Ce regroupement utilise un algorithme de classification hiérarchique ascendante. Le paramétrage de l'algorithme permet de contrôler de plus ou moins grand nombre de regroupements.

L'algorithme est illustré sur l'exemple d'un écosystème n'ayant que trois espèces : $sp0 \rightarrow sp1 \rightarrow sp2$. L'espèce *sp0* est la proie de l'espèce *sp1* et l'espèce *sp2* est le prédateur

de l'espèce *sp1* (les flèches représentent le transfert de biomasse). Les deux espèces *sp0* et *sp2* sont dites voisines de l'espèce *sp1*. Nous nous focalisons dans la suite sur la génération de l'automate de l'espèce *sp1*.

3.5.2.1 Construction de l'automate espèce par espèce.

La construction de l'automate d'une espèce passe par la construction des états qualitatifs stables, puis la construction des états intermédiaires, et finalement la construction des transitions étiquetées par des conditions et des contraintes temporelles.

L'algorithme de construction automatique des automates est schématisé dans la figure 3.4.

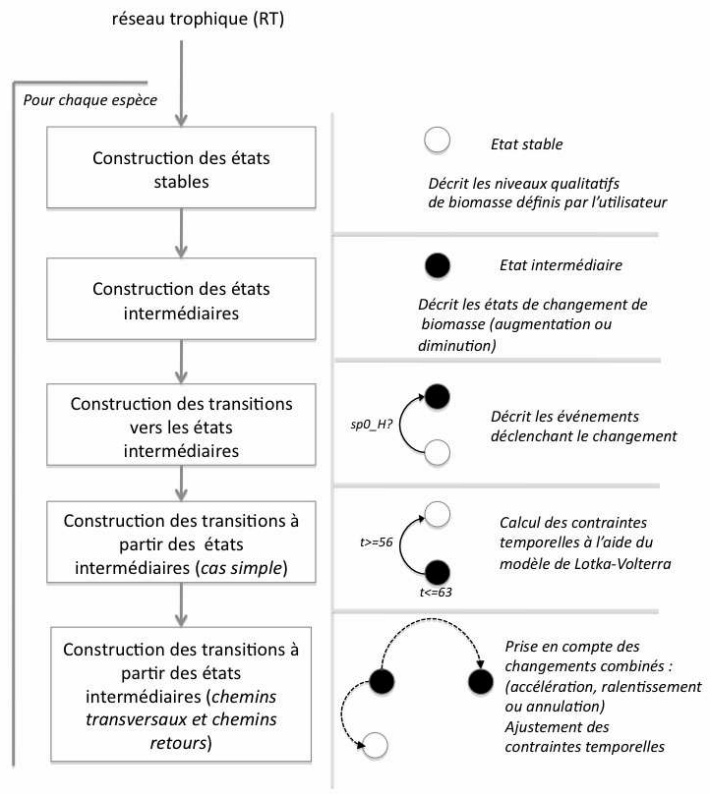


FIG. 3.4 – La génération automatique des automates se fait espèce par espèce et comprend quatre étapes principales. La figure illustre le lien entre les étapes et leurs effets (ajout d'états, de transitions, de contraintes) sur les automates en cours de construction

- *Construction des états stables :* chaque espèce est à l'équilibre dans un état qualitatif stable. Ces états sont décrits par des valeurs qualitatives, dans le cas celles de la biomasse de l'espèce. Pour l'espèce *sp1*, les états stables sont : *sp1_H, sp1_N, sp1_L*, correspondant aux trois valeurs qualitatives de biomasse, *H* (*High* pour élevé), *N* (*Normal*), et *L* (*Low* pour bas).

– *Construction des états intermédiaires :* lorsqu'un changement d'état qualitatif se produit dans une espèce voisine de *sp1* (dans le réseau trophique), ici *sp0* ou *sp2*, l'espèce *sp1* peut changer d'état qualitatif, en passant par un état intermédiaire, représentant la période d'évolution entre deux états stables. Il faut donc identifier toutes les combinaisons d'états qualitatifs des espèces voisines, qui peuvent faire évoluer l'espèce et la mener dans un tel état intermédiaire. À partir de l'état stable *sp1_N* par exemple, le tableau 3.2 présente 4 possibilités parmi 9, en indiquant la tendance (stable, croissante, décroissante) de l'évolution de la biomasse d'un état stable vers un autre, et sa durée. Un état intermédiaire est créé dans tous les cas où la tendance n'est pas "stable".
La première ligne du tableau 3.2 correspond au cas où soit *sp0*, soit *sp2* arrive dans l'état qualitatif *L (Low)* alors que l'autre y est déjà[1]. La tendance pour *sp1* est alors de rester stable et aucun état intermédiaire n'est créé. La troisième ligne correspond au cas où *sp0* arrive dans état qualitatif *H (High)* alors que *sp2* était dans l'état qualitatif *L (Low)*, ou celui où *sp2* arrive dans l'état qualitatif *L (Low)* alors que *sp0* était dans l'état qualitatif *H (High)*. La tendance est alors croissante pour *sp1*, avec une durée d'évolution entre *N (Normal)* et *H (High)* comprise entre 23 et 36 unités de temps. Un état intermédiaire est donc créé, sous le nom *sp1_N_sp0_H_sp2_L_UP*, obtenu par concaténation des noms des états des espèces et de la tendance.

État qualitatif départ de *sp1*	États qualitatifs des espèces voisines	Tendance	Durée min	Durée max
sp1_N	*sp0_L sp2_L*	stable	-	-
sp1_N	*sp0_L sp2_H*	décroissante	56	62
sp1_N	*sp0_H sp2_L*	croissante	23	36
sp1_N	*sp0_H sp2_N*	croissante	56	63

TAB. 3.2 – États intermédiaires envisagés et leurs caractéristiques

– *Construction des transitions vers les états intermédiaires :* après avoir construit les états intermédiaires, examinons maintenant comment construire les transitions menant à ces états intermédiaires.
Ces transitions correspondent au changement de biomasse d'une espèce voisine, qui provoque l'évolution de la biomasse de l'espèce considérée, et donc un changement d'état stable vers l'état intermédiaire. L'état intermédiaire *sp1_N_sp0_H_sp2_L_UP* peut être la cible de deux changements d'état qualitatif, l'un provoqué par le changement de biomasse de *sp0* et l'autre par le changement de biomasse de *sp2*. Il s'agit des deux cas suivants : la proie *sp0* entre dans l'état qualitatif "*High*" alors que le prédateur *sp2* est dans l'état "*Low*", ou le prédateur *sp2* entre dans l'état qualitatif "*Low*" alors que la proie *sp0* est dans l'état qualitatif "*High*". On construit donc deux transitions, arrivant toutes deux dans l'état intermédiaire *sp1_N_sp0_H_sp2_L_UP*, étiquetées par l'événement de synchronisation dé-

[1] On suppose toujours que deux transitions ne peuvent pas être simultanées.

clenchant la transition, auquel on ajoute un *?*, et par la propriété qui doit rester vérifiée, appelée *garde*, soit :
transition 1 : *sp0_ H ?*, garde : *sp2==L* et transition 2 : *sp2_ L ?*, garde : *sp0==H*

– *Construction des transitions à partir des états intermédiaires (cas simple) :* il faut ensuite construire les transitions sortant des états intermédiaires vers l'état stable d'arrivée. Le cas simple est celui où l'évolution de la biomasse se produit sans qu'aucun autre changement ne perturbe le processus. L'algorithme fait de nouveau appel aux équations de Lotka-Volterra pour calculer les contraintes temporelles associées, ce qui permet d'obtenir l'invariant de l'état intermédiaire et les contraintes temporelles de la transition vers l'état stable d'arrivée. Cette transition est associée à l'envoi d'un message de synchronisation, permettant de signaler le changement d'état qualitatif de cette espèce aux autres espèces voisines du réseau trophique ; la transition est alors étiquetée par un événement de type synchronisation terminée par un *!*.
Si on se reporte à la troisième ligne du tableau 3.2, une transition est créée qui sort de l'état intermédiaire *sp1_ N_ sp0_ H_ sp2_ L_ UP* et arrive à l'état stable *sp1_ H*. On calcule l'invariant *t<=36* et on en étiquette l'état intermédiaire, puis on ajoute la synchronisation *sp1_ H !* et la contrainte *t>=23* à la transition sortante.

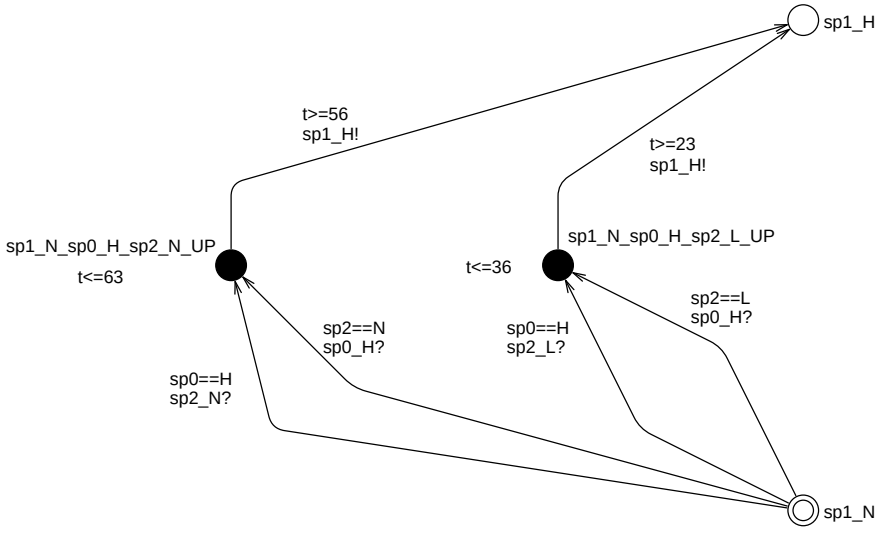


FIG. 3.5 – Ajout des transitions vers les états intermédiaires

La figure 3.5 illustre une partie de l'automate généré à la fin de ces quatre premières étapes. Les états blancs représentent les états qualitatifs stables de la biomasse de *sp1*. Les états noirs, dits états intermédiaires, représentent l'évolution de la biomasse de *sp1*. Les contraintes temporelles (t<=36 et t<=63) sur les états noirs représentent la durée maximale de l'évolution. Les transitions partant des états noirs et se dirigeant vers les états blancs correspondent à la fin de l'évolution. Ces transitions sont étiquetées par une contrainte temporelle représentant

la durée minimale de l'évolution et par un éventuel événement de synchronisation (terminé par un !). Les transitions partant des états blancs et se dirigeant vers les états noirs correspondent au début d'une évolution. Les contraintes sur ces transitions sont les conditions de déclenchements.

– *Construction des transitions à partir des états intermédiaires (cas des chemins transversaux et des chemins de retour) :* lorsqu'une espèce est dans un état intermédiaire, elle y reste tant que les contraintes temporelles identifiées à l'étape précédente n'exigent pas d'en sortir. Ces contraintes temporelles expriment le temps nécessaire au passage entre deux états stables voisins, en absence de tout autre événement. Or, des changements sur les autres espèces peuvent accélérer, ralentir ou même interrompre l'évolution en cours, s'ils interviennent pendant cet intervalle de temps. Pour représenter ces interactions, il est nécessaire d'ajouter un certain nombre de transitions, permettant, soit le retour vers l'état stable avant le début de l'évolution en cas d'interruption, soit la transition vers des états dits transversaux, intermédiaires eux aussi, qui permettent d'exprimer l'accélération ou le ralentissement de l'évolution par de nouvelles contraintes temporelles.
Puisque l'on ne peut passer en un seul changement que vers un état stable voisin, il suffit, à partir de l'état intermédiaire *sp1_N_sp0_H_sp2_L*, de rechercher l'existence des états intermédiaires : *sp1_N_sp0_H_sp2_N* et *sp1_N_sp0_N_sp2_L*. Si ces états intermédiaires ont déjà été construits lors de l'étape précédente, on ajoute une transition (chemin transversal) entre les deux états intermédiaires. C'est le cas du premier état intermédiaire *sp1_N_sp0_H_sp2_N*, et la nouvelle transition (en pointillé sur la figure 3.6) correspond à une évolution toujours croissante mais ralentie. Si ces états intermédiaires n'existent pas, on ajoute une transition retour qui ramène à l'état stable de départ. C'est le cas du second état intermédiaire *sp1_N_sp0_N_sp2_L*, et la nouvelle transition (en pointillé sur la figure 3.6) correspond à une interruption de l'évolution et ramène dans un état stable. Cette étape un peu technique est illustrée dans la figure 3.6. Les conditions étiquetant les transitions en pointillé ne sont pas données pour ne pas alourdir la figure.

3.5.2.2 Simplification des automates

Après avoir construit l'automate de chaque espèce, on procède à une simplification de ceux-ci.

Regroupement d'états intermédiaires : les états intermédiaires créés peuvent être nombreux et aussi être assez similaires. Or, le nombre d'états de l'automate augmente nettement la complexité de résolution des requêtes par model-checking. Il est donc intéressant de regrouper des états, en particulier ceux qui sont similaires. On utilise pour cela un algorithme classique de classification hiérarchique ascendante. Cet algorithme regroupe automatiquement les états en groupes d'états similaires (ou *clusters*) en s'appuyant sur une "distance". La distance d est calculée par la formule suivante : $d = \sqrt{\Delta I^2 + \Delta G^2}$ où ΔI (resp. ΔG) est la différence des invariants (resp. gardes) entre les états.

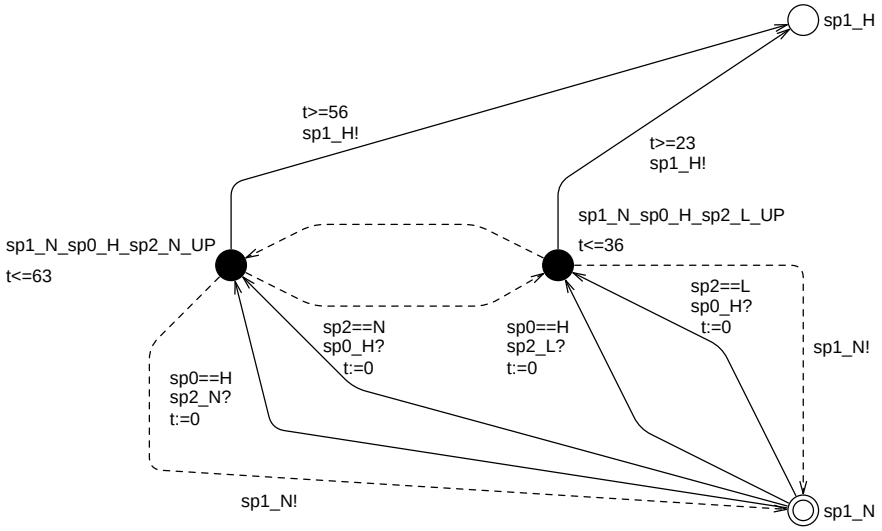


FIG. 3.6 – Construction des chemins transversaux et des chemins de retour. Les chemins en pointillé correspondent aux événements qui modifient la vitesse d'évolution (accélération ou ralentissement selon le sens) ou aux événements qui interrompent l'évolution en cours

Il est possible de contrôler le regroupement en indiquant le nombre maximal d'états intermédiaires que l'on autorise entre deux états stables voisins. Cette valeur est paramétrable par l'utilisateur. Plus on diminue ce paramètre, plus on perd en précision de la modélisation, mais plus on diminue la complexité de traitement d'une requête[2].

Regroupement de transitions : le regroupement d'états intermédiaires de l'étape précédente permet de supprimer certaines des transitions regroupées. C'est le cas lorsque les états intermédiaires regroupés vérifient les propriétés suivantes : les différences entre les états intermédiaires ne concernent qu'une seule espèce, et tous les états qualitatifs de l'espèce sont énumérés dans ce groupe.

C'est le cas par exemple lorsque les trois états intermédiaires suivants sont regroupés :

- $sp1\_N\_sp0\_L\_sp2\_H$
- $sp1\_N\_sp0\_N\_sp2\_H$
- $sp1\_N\_sp0\_H\_sp2\_H$

En effet, les différences entre ces états intermédiaires ne concernent que l'espèce *sp0* et $sp0\_L$, $sp0\_N$ et $sp0\_H$ sont toutes les valeurs qualitatives possibles de *sp0*. L'état qualitatif de *sp0* peut donc être ignoré et la condition d'entrée dans cet état intermédiaire ne dépend plus de *sp0* ; il est ainsi possible regrouper trois transitions en une seule.

À l'issue de l'optimisation de certains automates initialement complexes, le nombre

[2]Après une série d'expérimentations, la valeur 12 s'est avérée être un bon compromis, pour l'application, entre la précision du modèle et le temps de réponse des requêtes.

de transitions menant à un état intermédiaire est réduit de plus de 900 à moins de 300. Ceci réduit remarquablement la complexité du traitement des requêtes et en conséquence le temps d'interrogation.

3.6 Logiciel EcoMata

Nous présentons le logiciel EcoMata qui a été mon travail du stage de Master. Le but du travail était de développer une interface graphique permettant aux utilisateurs de saisir facilement la description d'écosystème et de lancer les requêtes de scénarios présentées dans la section 3.2 tout en intégrant le générateur d'automates précédemment développé en langage R. Après avoir analysé le code existant, j'ai décidé d'interpréter le générateur en langage Java pour une meilleur intégration puisque les composants graphiques du logiciel EcoMata ont été développés en Swing.

Cette section présente une vue globale du logiciel EcoMata et son utilisation sur l'exemple simplifié de l'écosystème. Le logiciel est composé de trois parties principales (cf. figure 3.7) :

- l'*éditeur d'écosystème* propose une interface graphique permettant aux utilisateurs de définir tous les éléments de l'écosystème : les espèces avec leurs paramètres biologiques, les pressions ainsi que leurs interactions ;
- le *générateur d'automate* construit, à partir de la définition de l'écosystème, d'une topologie par défaut des espèces et des pressions de pêche, le réseau d'automates temporisés au format d'Uppaal ;
- le *lanceur de requêtes* propose une interface graphique pour que l'utilisateur choisisse le patron de requête et définisse ses paramètres en entrée. Le lanceur interprète ce scénario en langage TCTL et exécute le model-checker d'Uppaal en fournissant le réseau d'automates et la formule de requête. À la fin de la vérification, le lanceur récupère les résultats et les traces d'exécution puis les affiche sous une forme textuelle ou graphique.

3.6.1 Éditeur d'écosystème

Pour illustrer la construction de l'écosystème à l'aide d'EcoMata, reprenons l'exemple du réseau trophique marin thon-maquereau présenté dans la section 3.3.2. Dans cet exemple, l'unité de temps est l'année mais le pas de temps peut être défini, au choix, par l'utilisateur. Pour chaque espèce de poissons, trois paramètres biologiques sont requis :

- B est la biomasse d'une espèce à l'équilibre. Cette valeur correspond à l'état qualitatif *Normal*. L'unité par défaut est $tonne/km^2$;
- P/B est le taux de croissance de la biomasse, en absence de prédateur et avec suffisamment de proies. Le taux est annuel ;
- Q/B est le taux de consommation de la biomasse. C'est la quantité d'aliment nécessaire pour que la biomasse d'une espèce puisse croître à la vitesse de P/B. Ce taux est également annuel.

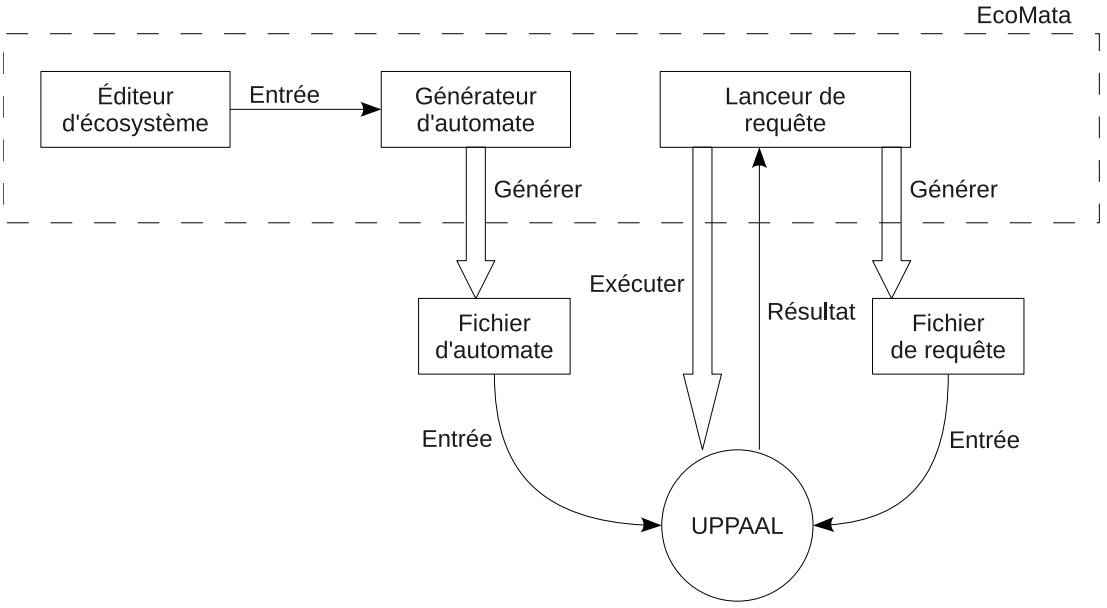


FIG. 3.7 – Architecture du logiciel : les trois composants et ses relations avec le model-checker UPPAAL. L'utilisateur entre le réseau trophique, les perturbations environnementales et les stratégies de pêche sur l'éditeur d'écosystème. Le générateur d'automate prend ces informations comme entrées et fournit en sortie un réseau d'automates modélisant l'écosystème. L'utilisateur lance ensuite des requêtes à l'aide des scénarios sur le lanceur de requête.

Les paramètres biologiques de l'écosystème thon-maquereau sont donnés dans le tableau 3.3. Pour un écologue, spécialiste d'un site d'étude, ces informations sont généralement bien connues. Le nombre d'états qualitatifs par défaut est de quatre (*Danger*, *Low*, *Normal* et *High*) mais l'utilisateur a la possibilité d'en définir le nombre qu'il souhaite. Chaque niveau correspond à une proportion du niveau *Normal* et est variable selon les espèces. Pour créer une nouvelle espèce, l'utilisateur peut, sur l'interface graphique, faire glisser une des espèces existant déjà dans la bibliothèque d'espèces ou la créer de toute pièce en saisissant les paramètres biologiques présentés ci-avant. Chaque nouvelle espèce créée peut enrichir la bibliothèque d'espèces ou être exportée afin d'alimenter éventuellement d'autres écosystèmes. Afin de rendre plus explicite le réseau trophique, les espèces sont représentées par des icônes dont les images sont laissées au libre choix de l'utilisateur. Dans l'éditeur d'écosystème, les pressions de pêche sont caractérisées par le pourcentage de biomasse capturée de l'espèce pêchée. Elles sont également définies par des niveaux qualitatifs, dont le nombre, la qualification et le rapport (au niveau normal) sont donnés par l'utilisateur.

Les interactions entre les espèces sont définies par le taux de prédation (DC) entre les prédateurs et les proies. Dans le logiciel, cette valeur est définie en cliquant sur le lien symbolisant le transfert de biomasse. Pour l'écosystème de l'exemple, les proportions

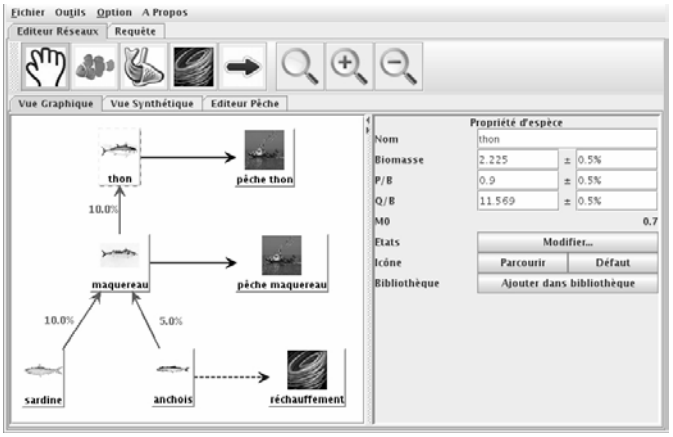


FIG. 3.8 – Création du réseau trophique de l'écosystème thon-maquereau à l'aide de l'éditeur d'écosystème d'EcoMata

Groupe	B $(tonne/km^2)$	$P/B(par\ an)$	$Q/B(par\ an)$
Thon	2,225	0,9	11,569
Maquereau	6,27	0,823	14,2
Sardine	13,79	2,74	14,0
Anchois	23,95	2,898	13,5

TAB. 3.3 – Paramètres biologiques de chacune des espèces de l'écosystème

de prédations sont les suivantes :

- 10 % de l'alimentation du thon est constituée de maquereau,
- 10 % de l'alimentation du maquereau est constituée de sardine et 5 % d'anchois.

Les autres sources d'alimentation des espèces représentées sont prises en considération dans le modèle écologique sous-jacent et ne doivent pas nécessairement être représentées dans le réseau trophique. En effet, il est souvent plus simple (et plus efficace du point de vue informatique) de limiter la définition de l'écosystème aux espèces touchées par la pêche ou la conservation.

Pour compléter la définition succincte de la pression de pêche donnée dans le réseau trophique, un onglet particulier appelé *éditeur de pêche* permet à l'utilisateur de définir précisément ses politiques de pêches dans le temps. Deux modes de définition des pressions de pêche sont possibles :

- schématisé : à l'aide d'un chronogramme, l'utilisateur définit la durée de la pression pour chaque niveau qualitatif de pression de pêche ;
- auto-adaptatif : la pression de pêche est guidée par le niveau de la biomasse. Ainsi pour chaque niveau atteint, on définit la pression de pêche correspondante (par exemple lorsque le niveau de la biomasse atteint un niveau *Low*, on réduit la pression de pêche).

L'intérêt de l'onglet *éditeur de pêche* est qu'il permet à l'utilisateur de réviser facilement ses politiques en fonction des résultats obtenus lors d'une précédente exécution. EcoMata peut alors être utilisé de manière interactive comme un outil d'aide à la décision.

Dans le réseau trophique thon-maquereau (cf figure 3.2), les pressions de pêche ainsi que la perturbation environnementale sont définies à l'aide du mode schématisé (ou chronogramme). À l'état qualitatif *Normal*, 20% de la biomasse est prélevée pour le thon et pour le maquereau. Ce taux est doublé à l'état qualitatif *High*. Le thon est pêché avec le niveau *High* pendant 12 années avant que la pêche ne soit arrêtée (état qualitatif *Stopped*). Le maquereau qui est pêché à un niveau *Normal* pendant les 12 premières années de la simulation et est pêché au niveau *High* à partir de la treizième année. Ce cycle de 24 années est ensuite reproduit à nouveau. La perturbation *Réchauffement* simule le phénomène d'*el niño* qui arrive une fois tous les 4 ans. Ce réchauffement cause une diminution de 5% de la biomasse d'anchois. La définition de l'évolution des niveaux de pressions (pêche, réchauffement) est présentée dans la figure 3.9.

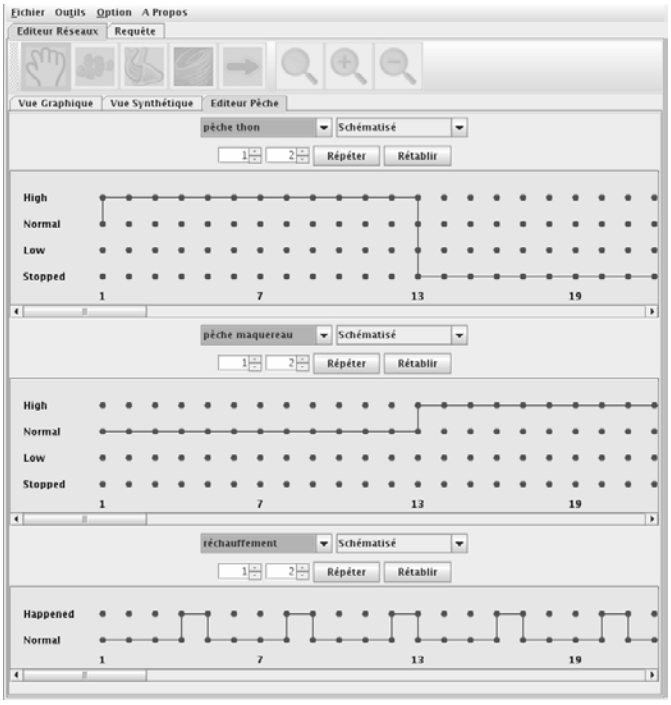


FIG. 3.9 – Définition des pressions de pêche à l'aide d'un chronogramme. Alternance des pressions aux niveaux *High* et *Stopped* pour le thon et *Normal* et *High* pour le maquereau

3.6.2 Générateur d'automates

Une fois le réseau trophique défini, le générateur peut être lancé pour construire un ensemble d'automates temporisés à partir des données du réseau trophique et de la topologie des espèces selon les niveaux qualitatifs définis [LC10]. Dans EcoMata, la représentation sous forme d'automates n'est pas directement accessible à l'utilisateur qui ne manipule que le réseau trophique, les chronogrammes de pressions de pêche ou les patrons de requêtes. Il est cependant nécessaire de générer le réseau d'automates pour résoudre les requêtes traduites en TCTL par model-checking. L'algorithme de génération de l'automate est décrit en section 3.5.

3.6.3 Lanceur de requête

L'algorithme de génération informe l'utilisateur de la complexité de l'automate généré et des délais attendus pour l'exécution de ses requêtes. Un menu déroulant permet d'accéder aux cinq patrons de requêtes ainsi qu'à une requête de synthèse (résumant les états qualitatifs de toutes les espèces sur la durée de la simulation). Cette section présente quelques exemples d'interrogation sur le réseau trophique thon-maquereau.

La première requête posée est : "Les stratégies de pêche définies sur le thon et le maquereau vont-elles conduire la biomasse du maquereau à l'état qualitatif *Danger* ?" La requête *Never* répond aux questions de type sûreté. Le résultat de cette requête, produit en quelques secondes, nous informe que cette situation "peut arriver" (cf. figure 3.10).

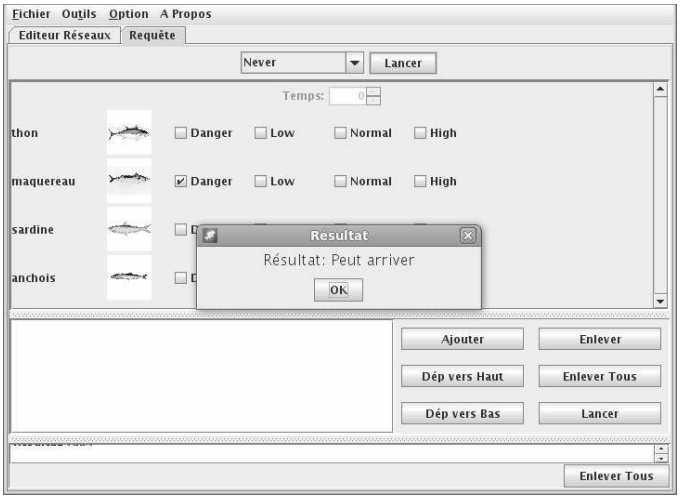


FIG. 3.10 – Requête de type "Never" invoquée pour une situation où le maquereau est à l'état qualitatif *Danger*

Étant donné la réponse précédente, on veut savoir à quelle date cette situation peut se passer au plus tôt. La requête *WhichDate*, pour laquelle, on coche simplement

comme paramètre d'entrée, le maquereau à l'état qualitatif *Danger*, nous informe que cette situation peut se produire en 26 ans (cf. figure 3.11).

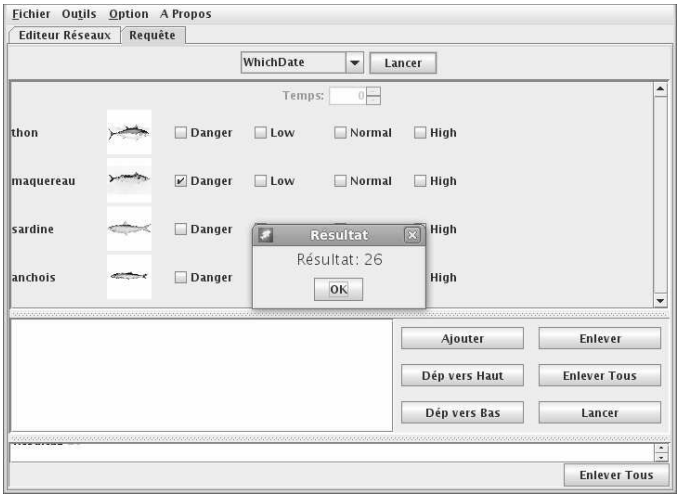


FIG. 3.11 – Requête du type "WhichDate" invoquée pour une situation où le maquereau est à l'état qualitatif *Danger*

Pour connaître l'état du système à cette période, la requête *Which-States* est lancée à la date 30. La figure 3.12 présente le résultat de cette requête sous la forme d'une liste de situations possibles. Un bouton au bout de chaque situation permet d'ouvrir une nouvelle fenêtre dans laquelle figure une trace possible menant à cette situation (voir la trace menant à la *situation 1* dans la figure 3.13).

FIG. 3.12 – Résultat de la requête "WhichStates" invoquée pour la date 30 années

3.7 Benchmarks (limites d'utilisation)

Afin d'évaluer les temps de réponses des requêtes soumises au model-checker d'UPPAAL, appelé VERIFYTA [LPY97], qui croissent avec la complexité des automates, nous nous sommes intéressés à la requête de type *Reachability*. Cette requête est la composante de base des patrons plus compliqués que sont *WhichStates* et *WhichDate*. Les requêtes *Never* et *Always* sont plus simples et ne prennent que quelques secondes quelle que soit la taille du système.

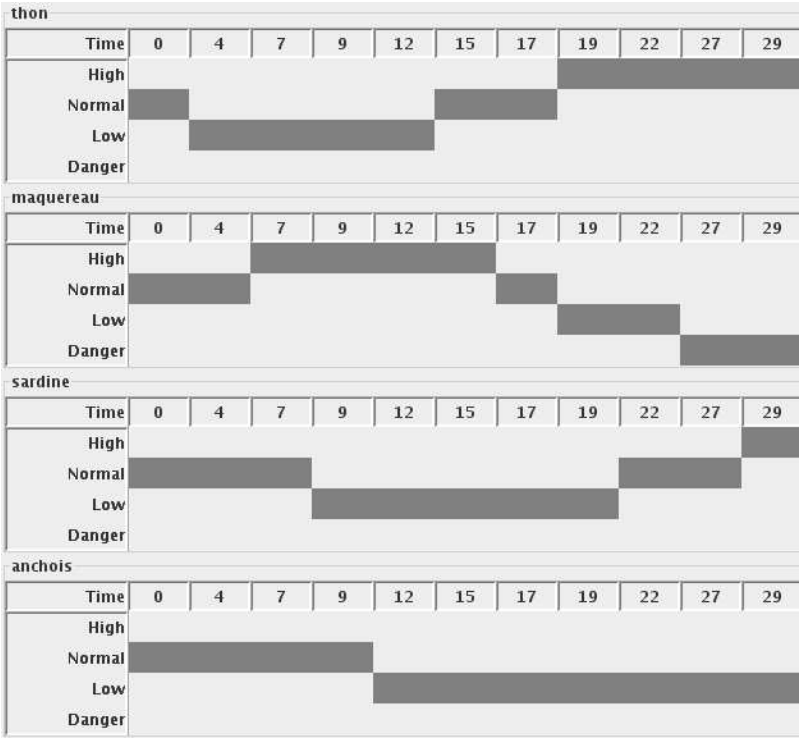


FIG. 3.13 – Trace menant à la *situation 1* (thon : *High*, maquereau : *Danger*, sardine : *High*, anchois : *Low*)

Parmi les diverses caractéristiques des automates, celles qualifiant la complexité du modèle global ont été exclusivement considérées. Ces caractéristiques sont, entre autres, les nombres d'états et de transitions du modèle (tous automates confondus) ainsi que le nombre d'états de l'automate le plus grand du réseau (en nombre d'états). Cette dernière variable n'augmente pas linéairement en fonction du nombre d'espèces du réseau trophique et dépend fortement du nombre d'interactions entre les espèces.

Pour pouvoir évaluer la complexité d'automates à partir des caractéristiques listées ci-dessus, huit type de réseaux trophiques ont été choisi (nombre d'espèces, nombre de pressions de pêche et les relations proie-prédateur entre les espèces). En variant les paramètres des espèces (biomasse, P/B et Q/B), 100 exemples de réseau ont été aléatoirement générés pour chaque type de réseau. Les caractéristiques et le temps de traitement de la requête de chaque exemple ont été relevés pour former une base d'apprentissage. La colonne temps de traitement est transformée en catégorique de la façon suivante :

- inférieur à 10 secondes : peu complexe ;
- entre 10 secondes et 1 minute : moyennement complexe ;
- supérieur à 1 minute : très complexe.

Un algorithme de l'arbre de décision du logiciel RAPIDMINER a été utilisé pour

Nombre d'espèces	Nombre de forces de pêche	Nombre d'états	Nombre de transitions	Nombre d'états de l'automate le plus grand	Temps de réponse *Reachability*
1	1	55	92	51	0,245
2	1	357	654	357	0,664
3	1	783	2146	717	2,090
4	2	836	2237	717	2,038
5	2	1184	2867	717	2,472
6	2	1644	4410	777	3,523
7	2	1902	4894	777	4,3
8	2	3484	9316	777	6,093

TAB. 3.4 – Résultats expérimentaux sur des réseaux trophiques caractérisés par le nombre d'espèces et le nombre de forces de pêche : nombre d'états et de transitions du modèle global, nombre d'états de l'automate le plus grand du modèle et temps de réponse pour une requête de type Reachability

générer une classification de la complexité d'automates. Cette classification a été implémentée dans EcoMata. À l'issue de la génération des automates, un message informe l'utilisateur de la catégorie de son modèle et, en conséquence, des temps de réponse estimés.

3.8 Conclusion

Ce chapitre présente EcoMata, une approche de modélisation qualitative d'un réseau trophique basée sur les automates temporisés. Le premier intérêt de cette approche est de proposer un formalisme unifié et qualitatif pour représenter tous les éléments constituant un écosystème. Cette approche générique permet de représenter un réseau trophique de type proie-prédateur, ainsi que les pressions anthropiques, les événements liés à l'environnement et tout autre entité interagissant avec l'écosystème, dans un même formalisme, celui des automates temporisés. Un avantage important de ce formalisme est de pouvoir représenter les contraintes temporelles ainsi que l'incertitude inhérente à l'évolution du stock d'une ressource. La discrétisation de la biomasse selon des niveaux usuels pour l'utilisateur permet une description générique de la topologie d'une espèce, ou d'une pression, qui autorise la construction automatique du réseau d'automates représentant l'écosystème. La génération automatique de l'automate, qui utilise le modèle numérique du Lotka-Volterra, repose sur un algorithme complexe afin de produire un ensemble d'automates suffisamment complet mais également exploitable dans la pratique.

Un autre intérêt de cette approche est de profiter de l'efficacité des techniques de model-checking pour répondre aux requêtes d'un utilisateur, non par simulation, mais par la vérification de propriétés sur l'écosystème. Cinq patrons de requêtes prédictifs

dans un langage de haut niveau ont été prédéfinis pour explorer le modèle. Ces patrons de requêtes proposés permettent une grande variété d'interrogations, puisqu'il est possible d'obtenir des informations sur les délais d'obtention d'une situation donnée, la possibilité d'atteindre une situation indésirable, les états possibles du système à une date donnée, la continuité d'une propriété au fil du temps, etc. Étant donné la modélisation choisie, les réponses aux requêtes sont généralement qualitatives (pour les états) et donc directement interprétables par l'utilisateur qui est en général directement le décideur, dans le cas d'étude le gestionnaire des pêches par exemple.

Dans ce travail, j'ai réalisé le développement du logiciel prototype EcoMata qui implémente cette approche de modélisation. Le logiciel dispose d'une interface graphique sur laquelle les utilisateurs peuvent saisir la description de réseau trophique et lancer les requêtes pour explorer le réseau. J'ai aussi participé à l'implémentation et en particulier l'optimisation de l'algorithme de la génération d'automates qui permet de tranformer rapidement la description d'un réseau trophique à un ensemble d'automates temporisés. Puisque le temps de génération de l'automate est faible (de l'ordre de quelques secondes), il est possible d'engager un processus interactif, en modifiant le scénario de pêche, en regénérant un nouveau modèle, puis en interrogeant de nouveau le modèle généré.

D'un point de vue applicatif, une évaluation rétrospective a été effectuée, en travaillant sur une série de données concernant la Mer du Nord sur une période assez longue, afin de comparer les prédictions produites par EcoMata et les données réelles [Pai11]. Ce travail a montré que les simulations produites par EcoMata sont conformes aux données historiques. Ce travail a aussi permis d'analyser finement de la plus ou moins grande précision du modèle l'impact sur les résultats, cette précision résultant de la plus ou moins grande simplification lors de la génération automatique.

La suite de ce travail est le traitement des requêtes proactives du type "Que faire pour ?", par une approche utilisant la technique de la synthèse de contrôleur sur les automates temporisés [ABC+05]. L'idée est par exemple de donner au logiciel la possibilité de proposer des politiques de pêches garantissant des états spécifiques du système, comme par exemple atteindre les états souhaités, ou éviter les états critiques pour certaines espèces. Cette approche, qui a fait l'objet de mon travail de début de thèse, est présentée dans le chapitre suivant.

Chapitre 4

Recherche de stratégies de gestion de pêche

4.1 Introduction

Nous présentons, dans ce chapitre, l'extension du modèle EcoMata pour répondre à des requêtes dites proactives de type "Que faire pour ?", et en particulier les requêtes dites de sûreté de type "Que faire pour éviter telle situation ?". Nous présentons d'abord une approche s'appuyant sur la technique de synthèse de contrôleur. La synthèse de contrôleur est une technique, de type model-checking, permettant, à partir d'automates de jeux, dans lesquels on distingue actions contrôlables et actions non contrôlables, de fournir des règles de décision répondant à un objectif donné, par exemple un objectif de sûreté [AMPS98]. Après l'analyse des résultats, une autre approche plus pragmatique est présentée. Cette approche, aussi complète, utilise le model-checking dans une démarche de type "générer et tester".

Nous présentons les automates de jeux et la synthèse de contrôleur dans la section 4.2. La section 4.3 décrit la modélisation de l'écosystème en automates temporisés de jeux et le méta-modèle de pression de pêche qui sera manipulé par l'algorithme de synthèse de contrôleur. Nous présentons ensuite l'utilisation de l'approche synthèse de contrôleur et analysons les résultats obtenus avec le logiciel Uppaal-Tiga dans la section 4.4. Nous présentons également une approche alternative de type "générer et tester" ainsi les résultats obtenus en section 4.5 avant de conclure dans la section 4.6.

4.2 Synthèse de contrôleur

Dans cette section, nous présentons la théorie de la synthèse de contrôleur en commençant par la définition de l'automate temporisé de jeux.

4.2.1 Automate temporisé de jeux

Reprenons la définition de l'automate temporisé 3.2.1 :

Définition 4.2.1. Un *automate temporisé de jeux TGA* est une extension d'un automate temporisé dans laquelle l'ensemble d'étiquettes Act est séparé en deux catégories disjoints : Act_c et Act_u. Les transitions dont l'étiquette $\sigma \in Act_c$ représentent les *actions discrètes contrôlables* alors que les transitions dont l'étiquette $\sigma \in Act_u$ représentent les *actions discrètes incontrôlables* [CDF+05].

4.2.2 Algorithme de synthèse de contrôleur

Définition 4.2.2. On note $S = \{\langle q,\ v\rangle\}$ l'ensemble des *configurations de l'automate*. On note $\Sigma = Act_c \cup \lambda$ un ensemble d'actions où Act_c est l'ensemble d'actions discrètes contrôlables et λ est l'action d'attente (laisser le temps passer). Une *règle de décision* est une paire ordonné (s, σ) où $s \in S$ et $\sigma \in \Sigma$ qui signifie quelle action faire dans quel *configuration de l'automate*.

Définition 4.2.3. Une *stratégie* f est une relation partielle entre l'ensemble des *configurations de l'automate* S et l'ensemble d'actions Σ, $f \subseteq S \times \Sigma$ [CDF+05]. On peut aussi dire qu'une stratégie f est un ensemble de *règles de décision*.

Définition 4.2.4. Un *contrôleur* C_f sur un automate temporisé de jeux TGA est un système couplé avec TGA qui le contrôle selon une *stratégie* f. On note $(C_f \| TGA)$ l'automate TGA contrôlé par C_f.

Le problème de la synthèse de contrôleur est défini comme suit : étant donné une propriété Φ, existe-il un contrôleur C_f pour un automate temporisé de jeux TGA telle que $(C_f \| TGA) \models \Phi$? Et si oui, quel est-il ? La synthèse de contrôleur peut traiter des problèmes de sûreté et/ou d'atteignabilité selon la propriété Φ considérée.

- Dans le cas du problème de sûreté, étant donné un automate temporisé de jeux TGA et une propriété φ à éviter, la synthèse de contrôleur cherche une stratégie $f \subseteq S \times \Sigma$ telle que $(C_f \| TGA) \models \forall\Box\neg\varphi$.
- Dans le cas du problème d'atteignabilité, étant donné un automate temporisé de jeux TGA et une propriété φ à atteindre, la synthèse de contrôleur cherche une stratégie $f \subseteq S \times \Sigma$ telle que $(C_f \| TGA) \models \forall\Diamond\varphi$.

Dans tous les cas, si la stratégie f n'est pas vide, on peut conclure qu'un tel contrôleur C_f existe. La stratégie f permet de construire le contrôleur. Il est possible que telle stratégie n'est pas unique. Soit un automate TGA, une propriété Φ et l'ensemble de contrôleurs $C^* = \{C_f \mid (C_f \| TGA) \models \Phi\}$, deux types de contrôleurs peuvent être intéressants :

- Le contrôleur complet C_{fmax} tel que $\forall C_f \in C^*, f \subseteq fmax$ où $fmax$ est la stratégie complète qui contient toutes les *règle de décision*s assurant $(C_{fmax} \| TGA) \models \Phi$. Il est donc possible que plusieurs actions sont associées à une *configuration de l'automate* dans cette stratégie $fmax$. Le contrôleur C_{fmax} doit choisir une action à effectuer.
- Un contrôleur minimal C_{fmin} tel que $\neg\exists C_f \in C^*, f \subset fmin$ où $fmin$ est une stratégie minimale qui est un ensemble minimal de *règles de décision* assurant $(C_{fmin} \| TGA) \models \Phi$. Dans ce type de stratégies minimales, une seule action est

associée à une *configuration de l'automate* donné. La stratégie minimale n'est pas forcément unique. Les stratégies minimales sont en fait des fonctions partielles.

4.2.3 UPPAAL-TIGA

UPPAAL est un outil connu dans le domaine du *model-checking* disposant d'une interface graphique agréable à utiliser. UPPAAL-TIGA est une version dérivée d'UPPAAL spécialement conçu pour effectuer de la synthèse de contrôleur sur un réseau d'automates temporisés de jeux [CDF+05]. L'algorithme a une complexité linéaire en fonction de la taille des automates traités. UPPAAL-TIGA traite des requêtes exprimées sous forme d'expressions TCTL. Lorsqu'une requête est satisfiable, UPPAAL-TIGA fournit en sortie un contrôleur, et donc un ensemble de *règle de décision*s (aussi appelé stratégie) permettant de satisfaire la requête.

UPPAAL-TIGA prend en entrée un ensemble d'automates de jeux, une requête et des options. La syntaxe est proche de celle utilisée dans UPPAAL [LPY97]. Les automates sont obtenus en ajoutant un attribut sur les transitions, qui indique si une transition est contrôlable ou pas, afin de transformer un automate temporisé classique en un automate temporisé de jeux. Nous nous intéressons à un seul type de requêtes parmi les cinq proposés par UPPAAL-TIGA, les requêtes de sûreté. La syntaxe d'une telle requête est :

$$control : A\Box\ not(\varphi)$$

où φ est la propriété à éviter.

UPPAAL-TIGA peut fournir, selon l'option demandée, soit le contrôleur complet, qui a l'intérêt de couvrir tous les autres, et décrit toutes les *règles de décision* ; soit un des contrôleurs minimals, choisi aléatoirement, qui correspond à une stratégie minimale.

4.3 Modélisation et méta-modèle

Nous présentons dans cette section comment la modélisation existante à base d'automates temporisés a été étendue pour obtenir des automates temporisés de jeux. Nous montrons aussi qu'il est nécessaire de contraindre la synthèse de contrôleur et comment cela se fait grâce à l'utilisation d'un méta-modèle.

4.3.1 Automates temporisés de jeux pour modéliser l'écosystème

Comme expliqué dans le chapitre 3, le modèle global décrit la dynamique d'un écosystème composé par un réseau trophique (espèces de poissons), les perturbations environnementales et les pressions de pêche. Nous nous intéressons en particulier aux pressions de pêche qui sont les seules transitions contrôlables du modèle. Nous avons donc :

- des automates non contrôlables (leurs transitions ne sont pas contrôlables) : ce sont les automates représentant le réseau trophique et les perturbations environnementales. Nous y ajoutons les pressions de pêche sur certaines espèces de poissons lorsque l'on considère que leur politique est figée ;

– des automates contrôlables (leurs transitions sont contrôlables) : ce sont les automates représentant les pressions de pêche dont la politique est à synthétiser.

Nous obtenons ainsi un ensemble d'automates temporisés de jeux (voir section 4.2).

4.3.2 Méta-modèle de l'automate de pêche

Partant de la modélisation des pressions de pêche, et en particulier celles décrites dans [LC10], des contraintes portant sur les pressions de pêche, et donc sur les automates les décrivant, ont été identifiées :

– Il ne peut y avoir qu'un seul changement de pression de pêche (et donc une seule transition dans l'automate correspondant) par unité de temps.
– Les transitions de l'automate de pêche ne doivent pas être déclenchées trop fréquemment. Il doit donc y avoir un intervalle minimum de temps entre deux changements de pressions de pêche.
– Les changements de pressions de pêche doivent être graduels, et donc on ne peut passer que d'un état qualitatif à un état qualitatif "voisin". En supposant l'ordre *Stopped*, *Low*, *Normal*, *High*, on peut passer de *Normal* à *Low* ou à *High*, mais on ne peut pas passer de *High* à *Stopped* ou *Low* et réciproquement.

Ces contraintes sur les modèles de pressions de pêche doivent être satisfaites par tous les comportements que propose le contrôleur. Un méta-modèle a été conçu afin de les exprimer formellement. Les automates de jeux devront être des instances de ce méta-modèle.

La figure 4.1 montre un exemple d'automate de pêche modélisant une pression de pêche nommée *ef1*, discrétisée à trois niveaux seulement : *Stopped*, *Normal* et *High* et n'autorisant des changements que toutes les 6 unités de temps. Le formalisme utilisé dans la figure respecte la syntaxe de l'outil UPPAAL-TIGA qui a été utilisé (voir section 4.4). Les trois cercles représentent les états qualitatifs de la force de pêche. Une horloge t et une variable locale *lastChangeDate* sont associées à cet automate. Les étiquettes sur les transitions sont :

– une garde $t > lastChangeDate \;\&\&\; t \; mod \; 6 == 0$ assurant que la date de déclenchement de la transition est strictement ultérieure au dernier changement d'état qualitatif et le temps passé depuis le dernier changement est multiple de 6 ;
– un signal de synchronisation $ef1\_N!$ qui propage l'événement du déclenchement aux autres automates partageant cette étiquette ;
– une mise à jour de la variable $lastChangeDate := t$ mémorisant la date du déclenchement actuel qui empêche un futur déclenchement tant que l'horloge n'a pas suffisament avancé.

Enfin, l'absence de transition entre l'état *High* et l'état *Stopped* assure que l'on ne peut passer entre ces deux états qu'en passant par l'état intermédiaire *Normal* afin d'assurer un changement graduel.

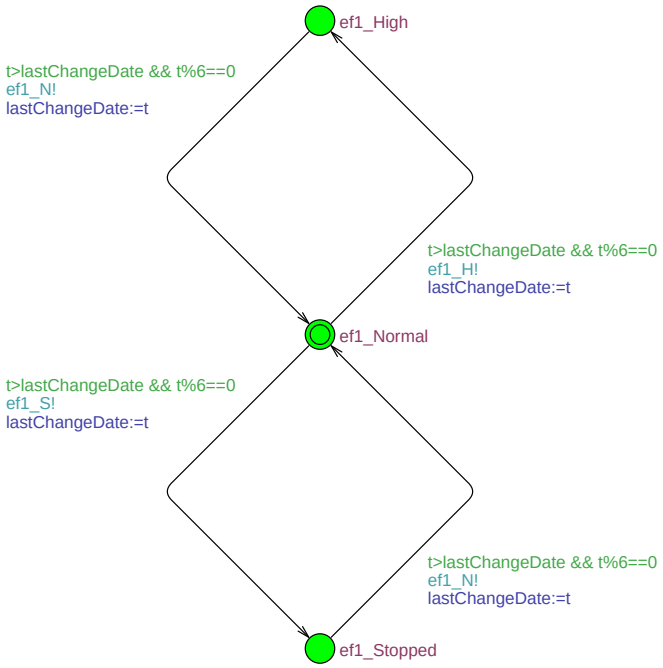


FIG. 4.1 – Un exemple de méta-modèle de pêche. Trois états qualitatifs : *Stopped*, *Normal*, *High*. Intervalle entre deux changements d'état qualitatif : multiple de six unités de temps.

4.4 Expérimentations et résultats

4.4.1 Résultats expérimentaux

Un réseau trophique de taille réduite a été choisi pour les premières expérimentations (cf. figure 4.2). Pour les politiques de pêche, les niveaux qualitatifs sont restreints à trois : (*Stopped*, *Normal* et *High*) et un intervalle multiple de 6 unités de temps (6 mois) entre deux changements d'état qualitatif a été imposé. Supposons que la politique de pêche *ef0* sur l'espèce *sp0* est figée (automate non contrôlable) et que la politique de pêche *ef1* sur l'espèce *sp1* est celle que l'on veut synthétiser.

La requête de sûreté exprimant que l'on veut éviter que la biomasse de *sp1* soit dans l'état qualitatif *Danger* s'exprime en logique TCTL par la formule :

$$control : A\Box\ not\ (sp1.sp1\_Danger)$$

Nous avons demandé à UPPAAL-TIGA de nous fournir un des contrôleurs minimals s'il en existe un. UPPAAL-TIGA a fourni une stratégie composée de 2120 *règles de décision*. Nous présentons quelques *règles de décision* de la stratégie proposée :

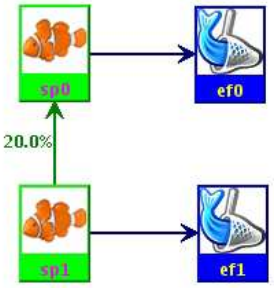


FIG. 4.2 – Ce réseau trophique est composé de deux espèces de poisson. *sp0* est soumis à la force de pêche *ef0* et *sp1* à la force de pêche *ef1*. Le pourcentage sur la flèche de *sp0* vers *sp1* indique que 20% de nourriture de *sp0* vient de la biomasse de *sp1*.

State : sp0_state=1, sp1_state=1, ef0_state=2
When you are in (chrono.t==0), take transition ef1.state_1->ef1.state_0

State : sp0_state=2 sp1_state=2 ef0_state=0
When you are in (chrono.t==72), take transition ef1.state_1->ef1.state_0

State : sp0_state=1 sp1_state=1 ef0_state=0
When you are in (chrono.t==48), take transition ef1.state_1->ef1.state_2

State : sp0_state=2 sp1_state=1 ef0_state=0
When you are in (chrono.t==24), take transition ef1.state_1->ef1.state_2

L'interprétation de la première *règle de décision* en langage naturel est : "Quand les espèces de poissons *sp0* et *sp1* sont dans l'état qualitatif où le niveau de biomasse est *Normal*, et le niveau de pression de pêche *ef0* est *High*, à l'unité de temps 0, diminuer le niveau de pêche de *ef1* à *Stopped*".

Après expérimentations, il est apparu un certain nombre d'inconvénients liées à l'utilisation d'UPPAAL-TIGA pour générer des politiques et répondre aux requêtes de sûreté :

- UPPAAL-TIGA est en version béta. Il n'est pas encore suffisamment stable et les plantages ne sont pas rares.
- La ressource en mémoire est limitée, et UPPAAL-TIGA ne peut traiter en conséquence que des modèles d'écosystèmes relativement simples.
- Le contrôleur généré est du type synchrone. Il faut connaître en temps réel l'état global du système pour évaluer les conditions et décider de l'action à effectuer. Cette caractéristique requiert de faire des observations en ligne ce qui est peu adapté dans un domaine environnemental où les capteurs sont en général rares et certaines mesures impossibles.
- Le grand nombre de *règles de décision* convient pour un contrôleur automatique,

mais ne convient pas dans un contexte d'aide à la décision, ou d'exploration par un utilisateur des différentes possibilités.
- Il faudrait intégrer une fonction d'évaluation dans la structure d'automate pour que UPPAAL-TIGA puisse proposer la meilleure politique ou ordonner les politiques possibles, mais ceci est difficile car le logiciel est relativement fermé.

Ceci explique l'approche alternative qui est présentée dans la section 4.5 n'ayant pas les inconvénients listés ci-dessus.

4.5 Synthèse par une approche "Générer et tester"

4.5.1 Motivations

Les difficultés exposées dans la section 4.4 ont amené à explorer une approche produisant des politiques exprimées sous une forme plus facilement exploitable par un utilisateur dans un contexte d'aide à la décision. Dans le domaine de la gestion de pêche, les politiques de pêche sont en général exprimées sous la forme d'un planning, encore appelé chronogramme, qui indique quels changements de pressions de pêche faire à quelle date. Ces politiques sont ainsi établies sur des critères temporels, et les décideurs n'ont pas à connaître le niveau de la biomasse des poissons pour prendre leur décision. La figure

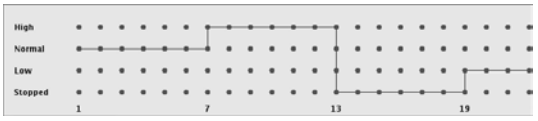


FIG. 4.3 – Planning de pêche sous forme de chronogramme

4.3 montre un exemple de chronogramme exprimé dans le logiciel ECOMATA [ZLC11] qui exprime sous une forme graphique les changements de pressions de pêche à des dates successives respectant l'intervalle de temps minimal fixé entre deux changements. Nous avons décrit dans le chapitre 3 l'algorithme qu'utilise ECOMATA pour transformer les entrées graphiques du logiciel, et en particulier le chronogramme, en un ensemble d'automates temporisés. Rappelons que ce logiciel permet, en utilisant le *model-checker* UPPAAL, de poser des requêtes et de récupérer les résultats de manière efficace et avec des temps de réponse raisonnables.

ECOMATA a été choisi comme le "vérificateur" pour construire et tester des politiques de pêche en suivant une approche "Générer et tester".

4.5.2 Génération et test des politiques

La synthèse basée sur l'approche "Générer et tester" se compose de trois étapes :

1. Énumérer tous les plannings possibles respectant les contraintes fournies en entrée telles que le nombre de niveaux de pressions de pêche considérés, la durée d'un plannings (nombre d'unité de temps du chronogramme), l'intervalle entre deux changements.

2. Trier les plannings générés selon une fonction d'évaluation qui ne dépend que de la nature du planning.
3. Tester les plannings dans l'ordre afin de déterminer si ils répondent aux critères exprimés dans la requête. On arrête le test dès qu'un planning satisfaisant aux critères de la requête est trouvé.

La fonction d'évaluation dans l'étape 2 peut prendre en compte des différentes critères ou préférences. Par exemple, on peut s'appuyer sur la facilité d'opération de pêche en privilégiant les planning avec le moindre nombre de changements. On peut également prendre en compte des facteurs économiques en favorisant les plannings maximisant la quantité de poissons pêchée pendant certaine saison.

4.5.3 Expérimentations et résultats

Pour pouvoir comparer cette méthode et la méthode proposée dans la section 4.4, le même écosystème (cf. figure 4.2) a été utilisé. Des politiques ont été générées pour la pression de pêche *ef1* selon quatre niveaux qualitatifs *High*, *Normal*, *Low* et *Stopped* et pour une durée de simulation de 36 unités de temps avec un intervalle de 6 unités de temps entre deux changements. Le nombre de politiques possibles p en fonction du nombre de niveaux qualitatifs des pressions de pêche n, de la durée de simulation d et de l'intervalle entre les changements i est :

$$p = n^{\lfloor \frac{d}{i} \rfloor}$$

Le nombre de politiques possibles dans notre exemple est donc $4^{\lfloor \frac{36}{6} \rfloor} = 4096$. La fonction d'évaluation de la politique que nous avons expérimentée est la quantité de biomasse prélevée qui estime le revenu de la pêche. Un poids est accordé à chaque niveau qualitatif, en sachant que les niveaux qualitatifs *High*, *Normal*, *Low* et *Stopped* représentent respectivement 2, 1, 0,5 et 0 fois la quantité de biomasse de l'état qualitatif *Normal*. Soit les unités de temps cumulés pour chaque niveau de pression de pêche t_H, t_N, t_L et t_S, la fonction d'évaluation S d'une politique est :

$$S = 2 \times t_H + 1 \times t_N + 0.5 \times t_L + 0 \times t_S$$

Nous avons choisi d'expérimenter quatre requêtes de sûreté :
- éviter que l'espèce *sp1* soit dans l'état qualitatif *Danger*
- éviter que l'espèce *sp1* soit dans l'état qualitatif *Danger* ou dans l'état qualitatif *Low*
- éviter que l'espèce *sp1* soit dans l'état qualitatif *Danger* ou que l'espèce *sp0* soit dans l'état qualitatif *Danger*
- éviter que l'espèce *sp1* soit dans l'état qualitatif *Danger* et que l'espèce *sp0* soit dans l'état qualitatif *Danger*

Ces expérimentations ont été faites en faisant varier la politique de *ef0* (supposée figée) pour tester la performance de l'algorithme. Les tables 4.1, 4.2 et 4.3 donnent les temps d'exécution pour chaque synthèse et les politiques trouvées. A noter que les chiffres dans la politique représentent les états qualitatifs de niveau de pêche se succédant sur une période de 6 mois avec 0-*Stopped*, 1-*Low*, 2-*Normal* et 3-*High*.

Comme on peut le voir sur les résultats, la synthèse termine dans la plupart de cas dans un délai raisonnable. Nous avons fait un essai de synthèse pour une durée de 48 unités de temps. Le nombre total de politiques à explorer passe de 4096 à 65536. Avec les mêmes requêtes, nous obtenons les résultats donnés dans la table 4.4.

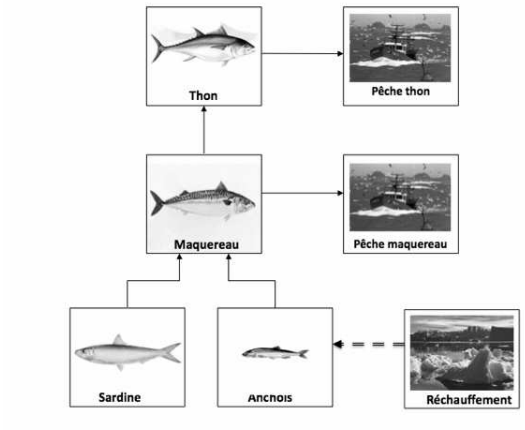


FIG. 4.4 – Réseau trophique thon et maquereau

Enfin, des expérimentations avec un modèle plus complexe (cf. figure 4.4) sur un écosystème différent ont été faites. Dans ce réseau trophique, le thon se nourrit du maquereau qui se nourrit de la sardine et de l'anchois. Le thon et le maquereau subissent chacun une pression de pêche et une perturbation environnementale, due au réchauffement de la planète, provoque une diminution régulière de la biomasse de l'anchois. On cherche la politique de pêche du maquereau, modélisée selon 4 niveaux, et pour une durée de 36 unités de temps, avec une intervalle de 6 unités entre deux changements. La politique de pêche du thon est considérée comme figée et est la suivante : 3 3 1 1 1 1.

- Pour la requête : non (*Maquereau Danger*), l'algorithme s'est arrêté sur la 184ème sur 4096 politiques après une demi-heure. La politique trouvée est 2 3 3 1 3 2.
- Pour la requête : non (*Thon Danger*), l'algorithme s'est arrêté sur la 31ème sur 4096 politiques après 7,6 heures. Parmi ces 31 vérifications, la plupart des vérifications ont une durée d'environ 10 minutes mais six vérifications se terminent par un plantage pour débordement de mémoire. La politique trouvée est 2 3 3 2 3 3.
- En raison de ces problèmes de plantage conduisant à des exécutions de longue durée, un mécanisme de minutage a été ajouté qui limite individuellement la durée de vérification à dix minutes. Au delà de cette borne, la vérification est abandonnée et est considérée comme non satisfaisante. Dans ce contexte, les vérifications se sont terminées à la 57ème sur 4096 politiques en 3,7 heures dont 19 vérifications dépassent la borne de 10 minutes. La politique trouvée est 2 3 3 1 3 3. Malheureusement ce mécanisme fait perdre la complétude théorique de l'algorithme.

Politique de *ef0* : 0 0 3 3 3 3				
Requête	Nb. politiques testées	Temps total d'exécution	Temps moyen par politique	Politique trouvée
non (*sp1 Danger*)	63 / 4096	44 secondes	695 ms	1 3 2 3 3 3
non (*sp1 Danger* ou *sp1 Low*)	1904 / 4096	21 minutes	686 ms	1 0 1 1 3 3
non (*sp1 Danger* ou *sp0 Danger*)	342 / 4096	163 secondes	679 ms	1 3 2 3 1 3
non (*sp1 Danger* et *sp0 Danger*)	63 / 4096	44 secondes	689 ms	1 3 2 3 3 3

TAB. 4.1 – Résultats de synthèse par l'approche "générer et tester" pour la pression de pêche $ef1$ sur l'espèce $sp1$, pour quatre requêtes, cas 1

Politique de *ef0* : 3 3 0 0 0 0				
Requête	Nb. politiques testées	Temps total d'exécution	Temps moyen par politique	Politique trouvée
non (*sp1 Danger*)	23 / 4096	16 secondes	701 ms	3 3 0 3 3 3
non (*sp1 Danger* ou *sp1 Low*)	3772 / 4096	42 minutes	686 ms	2 2 1 0 0 0
non (*sp1 Danger* ou *sp0 Danger*)	23 / 4096	16 secondes	679 ms	3 3 0 3 3 3
non (*sp1 Danger* et *sp0 Danger*)	1 / 4096	1 seconde	689 ms	3 3 3 3 3 3

TAB. 4.2 – Résultats de synthèse par l'approche "générer et tester" pour la pression de pêche $ef1$ sur l'espèce $sp1$, pour quatre requêtes, cas 2

Politique de *ef0* : 1 1 2 2 2 2				
Requête	Nb. politiques testées	Temps total d'exécution	Temps moyen par politique	Politique trouvée
non (*sp1 Danger*)	944 / 4096	11 minutes	681 ms	3 0 3 0 3 1
non (*sp1 Danger* ou *sp1 Low*)	4096 / 4096	47 minutes	690 ms	aucune
non (*sp1 Danger* ou *sp0 Danger*)	944 / 4096	11 minutes	700 ms	3 0 3 0 3 1
non (*sp1 Danger* et *sp0 Danger*)	1 / 4096	1 seconde	689 ms	3 3 3 3 3 3

TAB. 4.3 – Résultats de synthèse par l'approche "générer et tester" pour la pression de pêche $ef1$ sur l'espèce $sp1$, pour quatre requêtes, cas 3

Politique de *ef0* : 0 0 3 3 3 3 3 3				
Requête	Nb. politiques testées	Temps total d'exécution	Temps moyen par politique	Politique trouvée
non (*sp1 Danger*)	108 / 65536	93 secondes	863 ms	1 3 2 3 3 3 3 3
non (*sp1 Danger* ou *sp1 Low*)	22152 / 65536	311 minutes	842 ms	1 0 1 2 2 3 2 3
non (*sp1 Danger* ou *sp0 Danger*)	65536 / 65536	932 minutes	852 ms	aucune
non (*sp1 Danger* et *sp0 Danger*)	108 / 65536	92 secondes	853 ms	1 3 2 3 3 3 3 3

TAB. 4.4 – Résultats de synthèse par l'approche "générer et tester" pour la pression de pêche $ef1$ sur l'espèce $sp1$, pour une durée de 48 unités de temps

4.6 Conclusion

Ce chapitre présente une extension du modèle EcoMata afin de pouvoir traiter des requêtes de type "Que faire pour éviter telle situation ?", alors que n'étaient jusqu'alors traitées que les requêtes de type "Est-il possible d'atteindre telle situation ? À quelle date ?". L'objectif est d'étendre le logiciel EcoMata pour permettre à un utilisateur d'explorer un modèle qualitatif complexe et pour l'aider à prendre des décisions. Dans notre cas, une application a été choisie qui porte sur un écosystème marin, où l'évolution de plusieurs espèces de poissons dépend des interactions de type proie-prédateur, mais aussi des conditions environnementales (climatiques en particulier) et de la pression de pêche. Il est important de pouvoir déterminer les politiques de gestion de pêche permettant d'éviter des situations non souhaitables.

Deux approches ont été développées. La première approche s'appuyant sur la synthèse de contrôleur, issue du model-checking. Cette approche est générique et fournit un ensemble de *règles de décision* qui permet, quel que soit la configuration atteignable de l'automate dans lequel on se trouve, de satisfaire la propriété énoncée dans la requête. En revanche, l'approche dépend très directement des performances de l'outil utilisé, ici Uppaal-Tiga, et souffre aussi de ses limites (Uppaal-Tiga est encore en version bêta et manifeste une certaine instabilité). De plus, les stratégies obtenues sont plus adaptées à un contrôleur automatique qu'à un décideur tel qu'un exploitant agricole parce que (1) il faut pouvoir observer en ligne les états de tous les sous-systèmes, ce qui est impossible dans le domaine de halieutique et (2) les stratégies sont difficiles à utiliser par un être humain vu le nombre de *règles de décision* dans les stratégies obtenues.

Après avoir analysé les résultats de la première approche, nous avons présenté ensuite une autre approche, de type "Générer et tester", motivée par les difficultés rencontrées avec la première approche. Cette seconde approche est plus ad-hoc et utilise directement la plate-forme EcoMata précédemment développée, mais le mode d'expression des stratégies s'inspire de celui couramment employé par un gestionnaire d'écosystèmes, à savoir la planification (ici de gestion de pêche) par chronogramme. Ceci permet à l'utilisateur d'explorer les différentes options qu'il envisage dans un format qui lui est familier. Nous en avons présenté les principes, l'algorithme et analysé les résultats.

Après avoir expérimenté et comparé ces deux approches, il apparaît que la seconde approche répond mieux au problème de la génération de stratégies de gestion de la pêche. Cette nouvelle caractéristique a été ajoutée au logiciel EcoMata qui est disponible au site d'internet `http://oban.agrocampus-ouest.fr/ecomata`.

Troisième partie

Modélisation et recherche de stratégies optimales de gestion du pâturage

Dans cette partie de travail, nous nous intéressons à la recherche de stratégies optimales de la gestion du pâturage. Bien gérer les activités de pâturage est un défi car non seulement il faut bien comprendre les interactions entre la prairie et les troupeaux, mais aussi beaucoup de variables doivent être prises en compte, comme par exemple le climat. De plus les activités intensives d'élevage sont une des sources de pollution de nitrate dans un bassin versant. Des bonnes stratégies d'organisation de pâturage sont indispensables pour mieux utiliser la prairie et par ailleurs éviter d'apporter de la pollution à l'environnement. Nous avons choisi un outil de simulation de la gestion du pâturage PATUR'IN comme référence de la modélisation de pâturage et ceci nous a permis de travailler en interaction avec les experts du domaine et en particulier avec Luc Delaby, concepteur du logiciel qui a bien voulu nous confier le code source. Un des points limitants de PATUR'IN est qu'il ne dispose pas de mécanisme d'expression de stratégies de gestion du pâturage. Une des motivations de ce travail consiste à identifier de telles stratégies, de préférence optimales et accessibles à un utilisateur terrain.

Pour identifier des stratégies de gestion du pâturage, nous avons besoin tout d'abord d'un modèle de pâturage. Ayant vu l'intérêt de la modélisation par système à événements discrets d'un écosystème dans ECOMATA, l'idée est d'utiliser le même formalisme de modélisation dans le domaine du pâturage. En nous inspirant de la modélisation de pâturage du logiciel PATUR'IN, nous avons conçu un modèle par automates temporisés pour représenter les parcelles, la croissance de l'herbe et les actions de pâturage. Des expérimentations nous ont permis de voir que le modèle de croissance de l'herbe dans les parcelles requiert l'utilisation d'une fonction numérique. Nous avons, en conséquence, conçu une modélisation hybride qui couple un modèle numérique pour le calcul de croissance de l'herbe et un modèle en automates temporisés pour représenter les activités de pâturage. Grâce à l'outil de model-checking UPPAAL, la structure hybride du modèle de pâturage nous permet de bénéficier en même temps des avantages des deux types de modèle : la précision du calcul grâce au modèle numérique de l'herbe et la performance d'exécution grâce aux techniques de model-checking appliquées aux modèles en automates temporisés.

Pour mieux séparer les rôles dans le système, les automates sont organisés dans une hiérarchie de quatre couches. Un seul automate dans la couche *horloge centrale* joue le rôle de dirigeant du modèle. Il modélise l'avancement du temps et appelle les fonctions pour calculer la croissance de l'herbe. Les automates de la couche *contrôleur*, contrôlés par l'*horloge centrale*, modélisent les stratégies des activités de pâturage. Les automates dans la couche *exécution*, contrôlés par les automates *contrôleurs*, modélisent les activités de pâturage. Les modèles numériques dans la couche *prairie* représentent l'état de l'herbe dans les parcelles. Les modèles dans les couches *contrôleur*, *exécution* et *prairie* sont des méta-modèles. Pour pouvoir être utilisés, ils doivent être instanciés en fonction du nombre de troupeaux et du nombre de parcelles dans une exploitation.

Nous présentons, dans le chapitre 5, un logiciel prototype, nommé PATURMATA, implémentant cette modélisation hybride et hiérarchique qui permet de générer un ensemble d'automates temporisés à partir d'une description d'exploitation de pâturage et lancer des simulations. Le chapitre 6 présente les quatre méthodes que nous avons développées pour la recherche de stratégies optimales des activités de pâturage. La première

méthode s'appuie sur le logiciel PaturMata pour chercher la meilleure stratégie de mise au pâturage par énumération. La deuxième méthode utilise le formalisme des automates temporisés de coût. Cette extension des automates temporisés permet de représenter le coût du pâturage au long de l'exécution. Cette méthode utilise la technique de la synthèse de contrôleur sur les automates temporisés de coût pour chercher la meilleure stratégie de fertilisation, minimisant le coût du pâturage pour une exploitation de pâturage donnée. La troisième méthode consiste à chercher des stratégies génériques pour un type d'exploitations. Cette méthode utilise la technique de synthèse de contrôleur sur les automates temporisés de coût pour chercher des stratégies optimales de fertilisation pour un ensemble d'exploitations de même type. Un algorithme d'apprentissage automatique est utilisé pour générer des stratégies génériques. La quatrième méthode cherche à offrir plus de flexibilité à l'utilisation de stratégies car les stratégies générées par les autres méthodes ne le permettent pas. Cette méthode exprime les stratégies sous forme de méta-stratégies.

Chapitre 5

PaturMata - Modélisation de gestion de pâturage en automates temporisés

5.1 Introduction

Un certain nombre de systèmes de production laitière s'appuient principalement sur le pâturage et donc sur une alimentation du troupeau par l'herbe, impliquant une rotation des animaux sur les parcelles en prairie (déplacement des animaux d'une parcelle à une autre), avec éventuel complément alimentaire par du fourrage conservé et des concentrés lorsque le stock d'herbe sur les prairies n'est pas suffisant. L'objectif général est de maintenir le niveau de production laitière à un bon niveau, par une bonne utilisation de la prairie malgré des facteurs non-contrôlables tel que le climat. Les éléments que les exploitants peuvent manipuler sont le stock d'herbe sur les parcelles, la quantité de complément alimentaire offert aux animaux, la fertilisation azotée des parcelles et, le plus important, le choix de la parcelle à pâturer, modélisant des contraintes de déplacement du fait des aller-retours des animaux entre le bâtiment de traite et la parcelle à pâturer. L'idéal est que la production d'herbe soit la plus proche possible de la consommation d'herbe par les animaux. Une production élevée d'herbe nécessite des fauchages qui demande des ressources humaines et l'utilisation de machines. Une production insuffisante d'herbe nécessite de la consommation de fourrage conservé. En particulier, pour les exploitations avec une petite surface de prairies, l'utilisation d'une fertilisation azotée est courante pour stimuler la production d'herbe. Or, l'utilisation en excès de fertilisation pose des problèmes environnementaux. Une bonne organisation des activités de pâturage est ainsi nécessaire pour éviter les problèmes économiques et environnementaux.

Comme présenté dans la section 2.2, différentes approches d'aide à décision basées sur la simulation ont été développées pour explorer des stratégies de gestion de pâturage. Nous présentons dans ce chapitre une approche originale de modélisation du pâturage basée sur des automates temporisés. La section 5.2 présente la structure globale et les

caractéristiques du modèle. La section 5.3 présente en détail le modèle. Nous présentons ensuite dans la section 5.4 l'utilisation du modèle et quelques exemples de simulation. La section 5.5 présente un logiciel prototype basé sur cette approche. La section 5.6 décrit la validation et la calibration de la partie numérique du modèle. La section 5.7 présente une application du modèle avec données du bassin versant de Yar, situé en France, dans le département des Côtes d'Armor. Le chapitre se termine par une conclusion dans la section 5.8.

5.2 Modèle hybride hiérarchique de pâturage

EcoMata (voir le chapitre 3) propose une modélisation qualitative d'un réseau trophique du type proie-prédateur en automates temporisés. Dans cette modélisation, la biomasse des espèces est discrétisée en niveaux qualitatifs représentés par les états de l'automate. À partir de ces idées, le modèle qualitatif PaturMata a été créé avec deux caractéristiques supplémentaires : 1) une combinaison d'un modèle numérique et d'un modèle qualitatif ; 2) une structure hiérarchique. Cette section décrit ces caractéristiques en détail.

5.2.1 Modèle hybride

Les activités liées au pâturage que nous considérons sont la mise au pâturage, le fauchage et la fertilisation. Les autres activités liées aux troupeaux, tel que le vêlage, ne sont pas prises en compte car nous nous intéressons principalement aux impacts de différentes stratégies de gestion de pâturage en relation avec les conditions climatiques.

L'élément central de la gestion de la prairie est l'herbe. La hauteur de l'herbe est la donnée la plus importante, la plupart des décisions étant faites en fonction de sa valeur. Or, pour préserver la précision de calcul de la croissance de l'herbe avec une unité de temps journalière, un modèle en automate temporisé serait très complexe. Un modèle hybride a été créé qui combine un modèle numérique de la croissance de l'herbe et un modèle qualitatif en automates temporisés. Concrètement, la hauteur de l'herbe n'est plus modélisée en automate temporisé comme l'était la biomasse dans EcoMata. Des variables numériques sont utilisées pour représenter la hauteur de l'herbe dans chaque parcelle. Des fonctions numériques sont implémentées et servent à calculer et à mettre à jour la hauteur de l'herbe. Les activités liées au pâturage sont modélisées en automates temporisés. En conséquence, certains des automates temporisés du système utilisent les variables sur la hauteur de l'herbe dans leurs invariants et/ou leurs gardes alors que seules les horloges sont utilisées dans les automates temporisés d'EcoMata. D'une certaine manière, les variables sur la hauteur de l'herbe peuvent être considérées comme des horloges dans le modèle puisqu'elles jouent le même rôle.

5.2.2 Modèle hiérarchique

Comme montré dans la figure 5.1, le modèle est divisé en quatre couches : *Prairie*, *Exécution*, *Contrôleur* et *Horloge centrale* en fonction de leurs rôles dans le système.

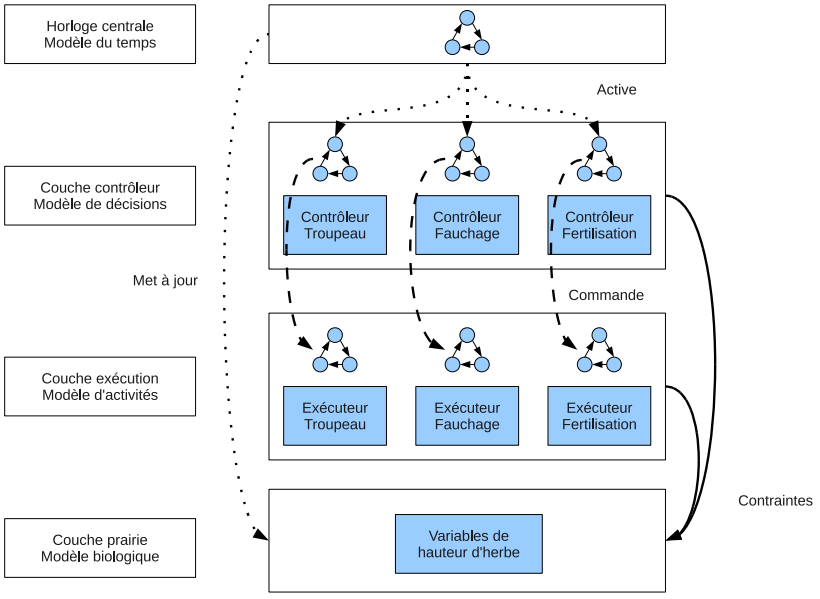


FIG. 5.1 – Structure du modèle hiérarchique

- Couche *Prairie* : Comme introduit dans la section 5.2.1, la hauteur de l'herbe est représentée par des variables numériques. Les variables sur la hauteur de l'herbe pour chaque parcelle et les fonctions qui calculent et mettent à jour ces variables constituent la couche *Prairie*. Cette couche est en fait la partie biologique liée à la production de biomasse pour l'alimentation animale. Elle fournit des informations aux autres couches.
- Couche *Exécution* : Les automates de la couche exécution modélisent les activités de pâturage dans une exploitation agricole. Trois activités principales ont été prises en compte : la mise au pâturage, le fauchage et la fertilisation. Chaque automate de la couche exécution possède au moins un état d'erreur. Les automates entrent dans ces états dès qu'ils reçoivent une commande qui viole les règles régissant les activités de pâturage. Les règles sont expliquées dans la section 5.3.2.
  - Les automates de mise au pâturage modélisent l'entrée et la sortie des animaux d'une parcelle et la consommation d'herbe et/ou de fourrage conservé pour ces animaux. Chaque automate représente un troupeau dans le système. Ces automates réagissent aux commandes de mise au pâturage en provenance de la couche contrôle grâce au mécanisme de synchronisation. Ils vérifient si les commandes reçues violent les règles de pâturage, par exemple, le fait qu'un troupeau est mis sur une parcelle déjà occupée par un autre troupeau. Ils surveillent également la hauteur de l'herbe pendant le pâturage et signalent quand la hauteur de l'herbe est en dessous du seuil de hauteur minimale.
  - Les automates de fauchage modélisent les activités de fauchage. Chaque automate correspond à une parcelle. Les automates de fauchage réagissent aux

commandes de fauchage en provenance de la couche contrôle. Ils vérifient, lors d'une demande de fauchage, que la hauteur de l'herbe est suffisamment haute pour démarrer un fauchage.
  - Les automates de fertilisation modélisent l'effet de la fertilisation sur les parcelles. Chaque automate correspond à une parcelle et réagit aux commandes de fertilisation en provenance de la couche contrôle. Chaque fois qu'une parcelle est fertilisée, l'effet de l'engrais dure une certaine période de temps.
- Couche *Contrôleur* : Les automates dans la couche contrôleur modélisent la prise de décision dans un système de pâturage. Ces automates contrôlent les automates dans la couche exécution grâce au mécanisme de synchronisation. Chaque automate de la couche contrôleur contrôle exactement un automate dans le couche exécution. Il y a donc autant d'automates de contrôle que d'automates d'exécution. Les automates de contrôle de fauchage et de fertilisation sont activés une fois par jour car deux actions successives de fauchage et de fertilisation ne sont pas possibles dans la même journée. Les automates de contrôle de mise au pâturage sont activés deux fois par jour : une fois pour décider s'il faut faire entrer le troupeau dans une parcelle si le troupeau est en dehors de parcelle, une deuxième fois pour décider s'il faut faire sortir le troupeau d'une parcelle si le troupeau pâture.
- *Horloge centrale* : L'horloge centrale est la modélisation de l'avancement du temps dans le système de pâturage. Elle active les automates de la couche contrôleur l'un après l'autre. Elle appelle les fonctions pour calculer et mettre à jour la hauteur de l'herbe dans les parcelles. Elle comptabilise le nombre de jours depuis le début d'exécution pour déterminer quand terminer l'exécution.

5.3 PaturMata - Instanciation du modèle

Les méta-modèles décrits dans la section 5.2 doivent être instanciés pour décrire la gestion de pâturage d'une exploitation. Les modèles instanciés s'inspirent du modèle numérique de pâturage PATUR'IN décrit en 2.2 et s'appuient sur l'expertise du domaine qui s'y trouve.

5.3.1 Couche prairie

La couche prairie modélise la dynamique de l'herbe sur les parcelles. Les parcelles sont caractérisées par les attributs numériques suivants : la surface, la densité d'herbe, le profil de croissance de biomasse d'herbe, la distance entre la parcelle et le bâtiment de traite et la hauteur de l'herbe. Deux attribut, l'accessibilité et l'hydromorphie du sol de parcelle, qui ne sont pas utilisés dans PATUR'IN ont été ajoutés dans PATURMATA.

- Le profil de croissance n'est pas une valeur constante pendant l'année. C'est un tableau qui contient 36 valeurs décadaires pour une année. Ces valeurs décrivent la croissance journalière de la biomasse de l'herbe. Elles dépendent du type d'herbe. La biomasse d'herbe sur une parcelle est définie par l'équation suivante : $BiomasseHerbe = HauteurHerbe * Densité * Surface$.

- La distance sert principalement à décider du choix de la parcelle pour une mise au pâturage d'un troupeau afin de diminuer la distance aller-retour de marche des animaux.
- L'accessibilité est un planning décadaire décrivant la restriction de l'utilisation de la parcelle. Pendant une décade marquée non accessible, il est interdit de mettre les animaux sur la parcelle même si la hauteur de l'herbe est bonne pour commencer un pâturage.
- L'hydromorphie du sol décrit la modification du profil de croissance d'une parcelle. Il y a un lien direct entre l'hydromorphie du sol et l'accessibilité d'une parcelle. Un sol trop humide diminue l'accessibilité à la parcelle.

Les fonctions de calcul relatives à l'herbe prennent en compte deux phénomènes : la croissance et la diminution de l'herbe. La partie de calcul pour la croissance prend cinq paramètres pour déterminer la croissance journalière de l'herbe : les conditions climatiques, le taux de croissance de biomasse de l'herbe, la biomasse de l'herbe présente sur la parcelle, la quantité d'engrais appliquée sur la parcelle et la présence d'animaux sur la parcelle.

- Les conditions climatiques sont issues d'un tableau qui contient 36 valeurs décadaires. Les valeurs dans ce tableau sont une parmi cinq valeurs discrètes : *Très favorable*, *Favorable*, *Normal*, *Défavorable* et *Très défavorable*. Ces valeurs servent à ajuster la croissance journalière de l'herbe sous une condition climatique donnée par rapport aux conditions climatiques qualifiée de *Normal*.
- Le taux de croissance de biomasse de l'herbe est une valeur du profil de croissance de l'herbe (kg MS/ha/jour), pour les conditions climatiques *Normal*.
- La biomasse d'herbe est une valeur au cours de l'exécution. Elle détermine la capacité de photosynthèse d'herbe et permet de calculer l'augmentation de la biomasse de l'herbe en combinaison avec le taux de croissance (kg MS/ha).
- L'engrais azoté stimule la biomasse de l'herbe en fonction de la quantité d'engrais appliqué et de la saison (kg N/ha).
- La présence d'animaux non seulement diminue la quantité d'herbe du fait de l'ingestion, mais aussi apporte un effet négatif sur la biomasse de l'herbe à cause de la défoliation due à l'ingestion des animaux. Concrètement, la biomasse de l'herbe en cas de présence d'animaux est la croissance normale multipliée par un ratio compris entre 0 et 1. Ce ratio évolue en fonction du nombre de jours où les animaux restent sur la parcelle. Plus les animaux restent longtemps, plus ce ratio est petit.

Le calcul de diminution prend en compte le nombre d'animaux et le besoin alimentaire journalier d'un animal, qui est déterminé par son poids et sa production laitière. Ce besoin est normalement satisfait par l'ingestion d'herbe de la prairie. Cependant, du fourrage conservé peut remplacer partiellement ou entièrement ce besoin d'herbe en cas de basse disponibilité d'herbe sur la prairie. De plus, l'utilisation de concentrés stimule la production laitière et réduit également le besoin d'herbe. Pendant l'exécution, le modèle de prairie calcule la quantité d'herbe ingérée en fonction du nombre d'animaux d'un troupeau et de l'ingestion journalière de fourrage conservé et de concentrés. La consommation journalière de fourrage conservé et de concentrés est discrétisée en

trois niveaux nommés *FourConsNiveau0*, *FourConsNiveau1* et *FourConsNiveau2* pour le fourrage conservé et trois niveaux nommés *ConcentréNiveau0*, *ConcentréNiveau1* et *ConcentréNiveau2* pour le concentré.

L'attribut de hauteur de l'herbe a été uniquement gardé dans le modèle en automates pour la facilité de calcul. Le profil de croissance de la biomasse de l'herbe est transformé en un profil de croissance de hauteur. Par exemple 100 kg MS/ha/jour devient 0,4 cm/jour, étant donné la densité de l'herbe est 250 kg MS/ha/cm. La consommation d'herbe par un troupeau en biomasse est aussi transformée en hauteur de l'herbe. Par exemple, une consommation de 18 kg MS/animal/jour de l'herbe fois 50 animaux sur une prairie de 2 ha devient 1,8 cm/jour, étant donné la densité de l'herbe est 250 kg MS/ha/cm.

5.3.2 Couche exécution

Trois activités principales du pâturage du modèle PATUR'IN ont été retenues et ont été implémentées dans PATURMATA : la mise au pâturage, le fauchage et la fertilisation. En tenant en compte de la pratique effective, des règles sont associées à chaque activité pour faciliter les opérations et/ou pour éviter certaines situations défavorables d'un point de vue environnemental. Les activités du pâturage pendant les exécutions ne doivent pas violer ces règles, sinon l'exécution est considérée comme invalide. Les règles sont paramétrables pour pouvoir modéliser différentes exploitations. Le tableau 5.8 indique tous les paramètres des règles et les valeurs choisies dans notre implémentation.

Les automates de la couche *exécution* modélisent ces trois activités de pâturage. Ils réagissent aux commandes envoyées par les automates de la couche contrôleur et détectent les violations de règles.

5.3.2.1 Modèle de mise au pâturage

Un automate de mise au pâturage décrit l'entrée et la sortie d'un troupeau d'une parcelle. Le troupeau peut être dans un des deux situations suivants : 1) être en dehors de prairie et rester dans le bâtiment de traite lorsqu'aucune parcelle n'est disponible pour la mise au pâturage ; 2) être au pâturage, dans une parcelle, avec ou non de l'ingestion de fourrage conservé et/ou concentrés.

La structure de l'automate de mise au pâturage est illustrée dans la figure 5.2. L'état *HorsParcelle* représente la situation du troupeau quand il est en dehors de la parcelle. Lors de la réception d'une commande *DébutPâturage* en provenance du contrôle de mise au pâturage, indiquant la parcelle où mettre le troupeau, le niveau de fourrage offert au troupeau pendant le pâturage et le niveau de concentré offert au troupeau, l'automate bascule dans l'état *Vérifier*. C'est un état transitoire représenté par un état engagé (couleur magenta avec une lettre C) qui signifie qu'il faut tester immédiatement si l'état de la parcelle satisfait les règles suivantes :

- La parcelle est accessible au moment de la réception de la commande *DébutPâturage*.
- La parcelle n'est pas occupée par un autre troupeau.

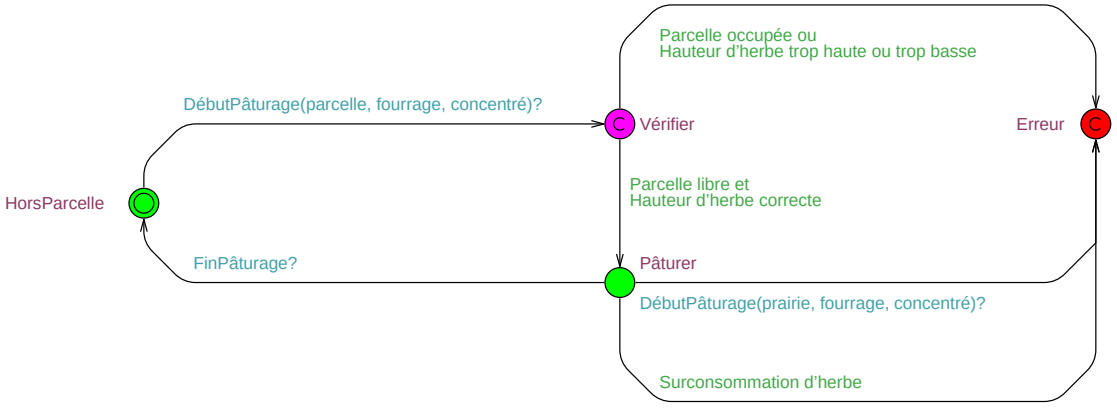


FIG. 5.2 – Automate de mise au pâturage

– La hauteur de l'herbe sur la parcelle est comprise entre *HauteurPâturageMin* et *HauteurPâturageMax*.

Si une de ces conditions n'est pas satisfaite, l'automate bascule dans l'état *Erreur*. Cet état est un état d'impasse qui interrompt immédiatement l'exécution. Concrètement l'état *Erreur* est un état engagé (voir la section 3.2.4) qui demande un départ immédiat sans laisser passer du temps. L'absence de transition sortant de cet état forme un blocage de l'exécution du modèle.

Si toutes les conditions sont satisfaites, l'automate entre dans l'état *Pâturer*. Il reste dans cet état jusqu'à recevoir une commande *FinPâturage*. Tant que l'automate reste dans l'état *Pâturer*, il vérifie une fois par jour l'état de la parcelle où le troupeau pâture. Lorsque la hauteur de l'herbe devient trop basse (inférieur à *HauteurFinPâturage*) et si les animaux sont toujours présents sur la parcelle, l'automate bascule dans l'état *Erreur*. L'automate bascule aussi dans l'état *Erreur* lorsqu'il reçoit une commande *DébutPâturage* alors qu'il est dans l'état *Pâturer*.

5.3.2.2 Modèle de fauchage

L'automate de fauchage décrit les activités de fauchage sur une parcelle. En tenant compte de la pratique effective, trois règles de fauchage ont été appliquées et implémentées :

– Pour la facilité de l'opération, il est impossible d'effectuer un fauchage quand la parcelle est occupée par un troupeau.
– Dû aux limites mécaniques de la machine de fauchage, la hauteur de l'herbe après un fauchage est fixée à une valeur constante définie par *HauteurAprèsFauchage*.
– Le fauchage nécessite de la consommation de l'énergie et de la ressource humaine. Pour des raisons économiques et environnementales, il est donc conseillé de ne déclencher un fauchage que si nécessaire. Cette règle dans notre modèle est définie par : un fauchage n'est possible que quand la hauteur de l'herbe dépasse *HauteurMinFauchage*.

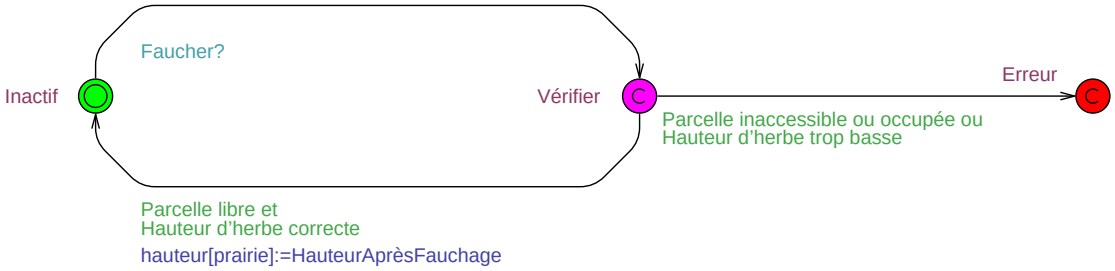


FIG. 5.3 – Automate de fauchage

Comme illustré dans la figure 5.3, lorsqu'il reçoit la commande *Faucher*, l'automate de fauchage bascule de l'état *Inactif* à l'état *Vérifier* représenté par un état engagé qui demande à vérifier immédiatement si l'état de la parcelle satisfait les règles. Si oui, l'automate retourne à l'état Inactif en mettant à jour la variable de la hauteur de l'herbe à *HauteurAprèsFauchage*. Sinon, l'automate bascule dans l'état impasse *Erreur*.

5.3.2.3 Modèle de fertilisation

L'automate de fertilisation décrit l'effet de la fertilisation d'une parcelle. Une fois l'engrais appliqué sur la parcelle, le taux de croissance de l'herbe est augmenté et l'effet de cette augmentation dure une certaine période de temps. Le cycle de l'effet de la fertilisation est décrit comme suit :

- Pendant $FD1$ jours à partir du jour où l'engrais est appliqué, l'herbe sur la parcelle absorbe l'engrais, mais le taux de croissance ne change pas pendant cette période.
- Pendant $FD2$ jours suivants (du $J+FD1+1$ au $J+FD1+FD2$), la fertilisation est effective. Le taux de croissance de l'herbe augmente en fonction de la quantité d'engrais et de la saison. Pendant cette période, quelle que soit l'activité sur la parcelle, l'effet de la fertilisation est présent.
- Pendant $FD3$ jours suivants (du $J+FD1+FD2+1$ au $J+FD1+FD2+FD3$), l'effet de la fertilisation continue. Mais dès que la parcelle est fauchée ou pâturée, l'effet de la fertilisation est interrompu.
- À la fin de $J+FD1+FD2+FD3$, l'effet de la fertilisation est fini. Le taux de croissance de l'herbe revient au niveau normal.

Quatre règles de fertilisation sont appliquées dans notre modèle en tenant en compte de la pratique effective :

- Pour la facilité de l'opération, il est impossible d'appliquer de l'engrais sur une parcelle si elle est occupée par un troupeau.
- Dû aux limites mécaniques de la machine, le montant d'une application d'engrais est fixé à une dose *DoseEngrais*.
- Pour les raisons environnementales, la quantité d'engrais cumulé sur une parcelle pendant une saison de pâturage (un an normalement) ne doit pas dépasser un seuil. Le paramètre *QuotaEngrais* représente le seuil pour les parcelles non hydromorphes et le paramètre *QuotaEngraisHydro* représente celui pour les parcelles

hydromorphes.

– Avant que l'effet de l'engrais appliqué sur une parcelle ne soit pas terminé, il serait possible d'appliquer une nouvelle dose d'engrais. Comme cette pratique n'est pas courante dans la réalité et afin de réduire la complexité des automates, une règle a été ajoutée dans le modèle qui interdit l'application d'engrais sur une parcelle lorsque l'effet de la dernière fertilisation n'est pas encore terminé. L'automate de fertilisation signale une erreur dès qu'une autre fertilisation est appliquée.

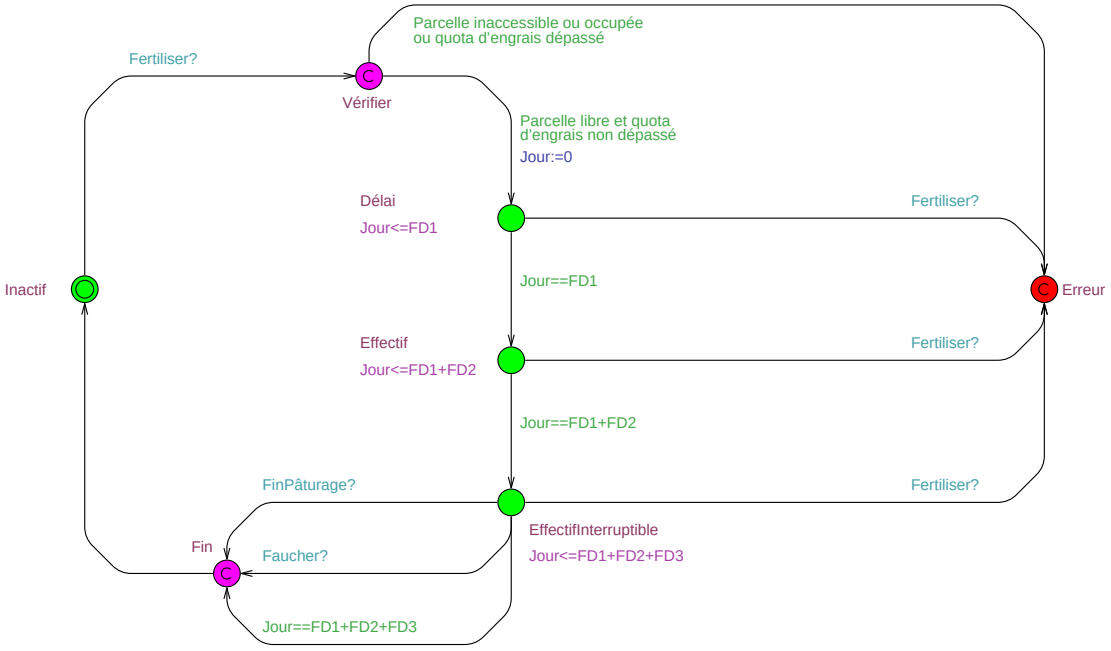


FIG. 5.4 – Automate de fertilisation

Comme illustré dans la figure 5.4, lorsqu'il reçoit une commande *Fertiliser*, l'automate de fertilisation bascule de l'état *Inactif* à l'état *Vérifier* qui est représenté par un état engagé. L'automate vérifie immédiatement que la parcelle n'est pas occupée par un troupeau et que le montant d'engrais cumulé ne dépasse pas le quota en fonction du type de parcelle. Si ces règles ne sont pas respectées, l'automate bascule dans l'état impasse *Erreur*. Sinon, l'automate débute le cycle de fertilisation en passant par les états *Délai*, *Effectif* et *EffectifInterruptible*. Si l'automate reçoit une autre commande *Fertiliser* quand il est dans un de ces trois états, il bascule dans l'état *Erreur* pour signaler une tentative de fertilisation quand l'effet de la dernière fertilisation n'est pas terminé. À la fin du cycle de fertilisation, l'automate retourne à l'état *Inactif*.

5.3.3 Couche contrôleur

Les automates dans la couche contrôleur sont conçus pour contrôler les automates de la couche exécution. Ils simulent la procédure de prise de décision basée sur l'état

de système de pâturage selon des stratégies de pâturage. Ils sont appelés chaque jour pendant l'exécution pour déterminer les actions à effectuer sur les parcelles. Les stratégies de pâturage décrivent les activités qu'il faut déclencher dans une situation précise. Elles expriment ce qu'il faut faire alors que les règles de pâturage expriment ce qu'il ne faut pas faire.

Nous avons choisi une structure générique des automates de contrôle pour chaque activité de pâturage. Dans cette structure, les stratégies des activités sont exprimées sous forme de fonctions dans un langage de programmation. Certains transitions des automates de contrôleur sont associées à ces fonctions. Quand ces transitions sont déclenchées, la fonction associée est appelée. Les automates de contrôle interprètent la valeur de retour de fonction et envoient la commande correspondante aux automates d'exécution. L'automate et les fonctions associées modélise une stratégie de pâturage.

Les avantages de cette combinaison sont :

- Nous n'avons pas à concevoir une structure d'automate de contrôleur propre à chaque stratégie.
- Certaines stratégies ne peuvent pas être exprimées sous forme d'automate temporisé alors que les fonctions sont mieux adaptées dans ces cas.
- La structure permet aux algorithmes de synthèse de contrôleur de générer des stratégies de pâturage quand les automates ne sont associés à aucune fonction.

Dans le modèle PaturMata, une seule stratégie de mise au pâturage est appliquée à tous les troupeaux. Une seule stratégie de fauchage et une seule stratégie de fertilisation sont appliquées à toutes les parcelles.

5.3.3.1 Modèle contrôleur de mise au pâturage

L'automate contrôleur de mise au pâturage contrôle un automate de mise au pâturage de la couche exécution. C'est-à-dire qu'il déclenche les actions de mise au pâturage pour un troupeau.

L'automate contrôleur de mise au pâturage est illustré dans la figure 5.5. L'automate reste dans l'état *HorsParcelle* quand l'automate de mise au pâturage qu'il contrôle est dans l'état *HorsParcelle* (cf. 5.3.2.1). Une fois activé par la réception d'une commande *DéciderDébutPâturage*, l'automate contrôleur bascule dans l'état *Choisir* qui est représenté par un état engagé où il appelle immédiatement la fonction de stratégie pour choisir une parcelle où mettre le troupeau en pâturage. Si une parcelle est choisie, l'automate de contrôleur envoie une commande *DéciderDébutPâturage* à l'automate qu'il contrôle et bascule dans l'état *Pâturer*. Sinon, il retourne à l'état *HorsParcelle* et attend l'activation suivante. Pendant que l'automate contrôleur reste dans l'état *Pâturer*, il est activé une fois par jour par la réception d'une commande *DéciderFinPâturage* pour décider s'il faut ou non enlever le troupeau de la parcelle. Si c'est le cas, il envoie une commande *FinPâturage* à l'automate qu'il contrôle et retourne à l'état *HorsParcelle*. Sinon, il reste dans l'état *Pâturer* et attend l'activation suivante.

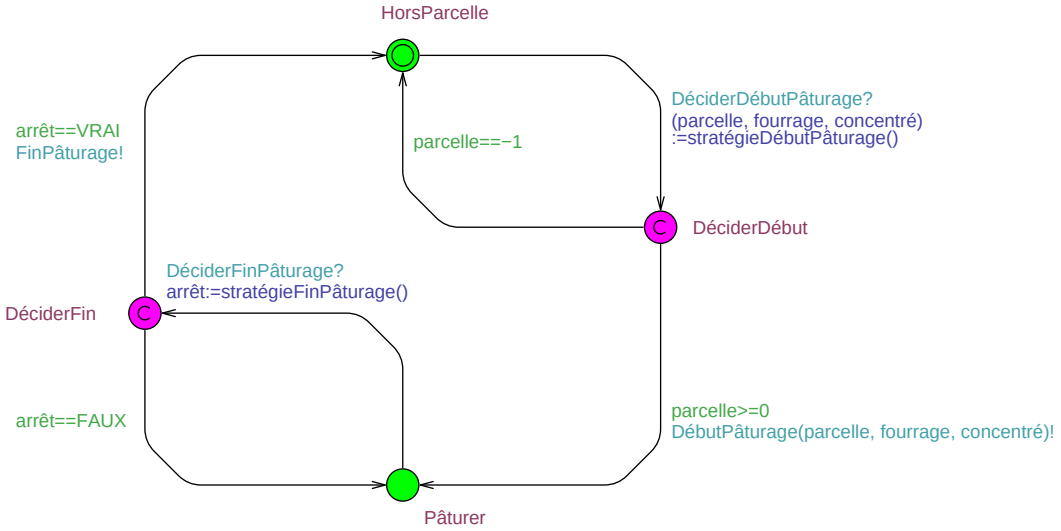


FIG. 5.5 – Automate contrôleur de mise au pâturage

5.3.3.2 Modèle contrôleur de fauchage

L'automate contrôleur de fauchage contrôle un automate de fauchage de la couche exécution. C'est-à-dire qu'il déclenche les actions de fauchage sur une parcelle.

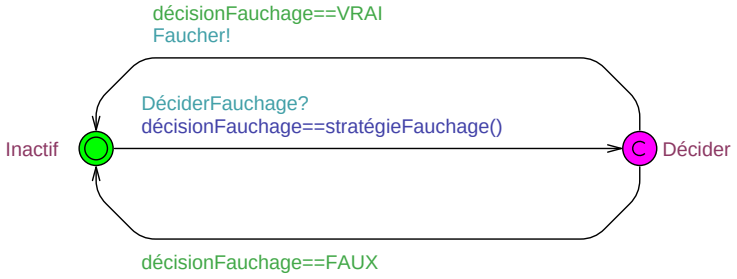


FIG. 5.6 – Automate contrôleur de fauchage

L'automate contrôleur de fauchage a deux états comme illustré dans la figure 5.6. Une fois activé, l'automate bascule de l'état *Inactif* à l'état *Décider* qui est représenté par un état engagé où il fait immédiatement une décision de fauchage en fonction de la stratégie de fauchage. S'il décide de faucher la parcelle, il envoie une commande *Faucher* à l'automate qu'il contrôle et retourne à l'état *Inactif*. Sinon, il retourne à l'état *Inactif* sans rien faire.

5.3.3.3 Modèle contrôleur de fertilisation

L'automate contrôleur de fertilisation contrôle un automate de fertilisation de la couche exécution. C'est-à-dire qu'il déclenche les actions de fertilisation sur une parcelle.

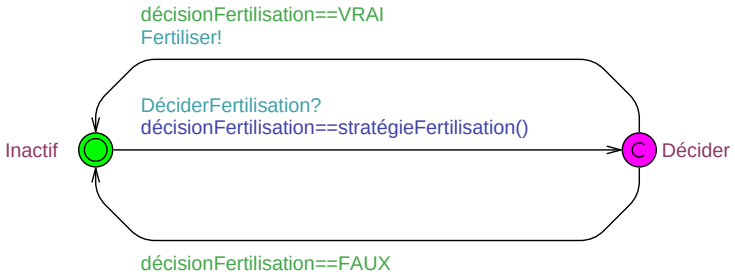


FIG. 5.7 – Automate de contrôleur de fertilisation

L'automate contrôleur de fertilisation a deux états comme illustré dans la figure 5.7. Une fois activé, l'automate bascule de l'état *Inactif* à l'état *Décider* qui est représenté par un état engagé où il fait immédiatement une décision de fertilisation en fonction de la stratégie de fertilisation. S'il décide de fertiliser la parcelle, il envoie une commande *Fertiliser* à l'automate qu'il contrôle et retourne à l'état *Inactif*. Sinon, il retourne à l'état *Inactif* sans rien faire.

5.3.4 Horloge centrale

L'horloge centrale modélise l'avancement du temps dans le système de pâturage. La structure de l'horloge centrale est cyclique puisqu'il répète la même tâche chaque jour (cf figure 5.8). Il commence par l'état *DébutJour*. Il active d'abord les automates contrôleurs de fauchage l'un après l'autre. Il active ensuite les automates contrôleurs de fertilisation. Les décisions de fauchage et de fertilisation doivent être faites avant de mettre les troupeaux dans les parcelles puisqu'il est interdit de faucher ni de fertiliser une parcelle quand elle est occupée par un troupeau.

Comme présenté dans la section 5.3.3.1, les automates contrôleurs de mise au pâturage peuvent être dans un des deux états *HorsParcelle* et *Pâturer* qui signifie respectivement que les animaux sont en dehors de la parcelle et que les animaux sont en pâturage sur une parcelle. L'horloge centrale active ensuite les automates contrôleurs de mise au pâturage qui sont dans l'état *HorsParcelle* pour faire des décisions de début de pâturage. L'horloge centrale arrive à l'état *MiseAJourPrairies* où elle appelle les fonctions de calcul afin de mettre à jour les variables de hauteur de l'herbe et effectuer des statistiques de pâturage. L'horloge centrale active ensuite les automates contrôleurs de mise au pâturage qui sont dans l'état *Pâturer* pour faire de décisions de fin de pâturage. Finalement l'horloge centrale met à jour la date et la saison. Une horloge d'automate temporisé est utilisée pour modéliser l'avancement du temps et la valeur de l'horloge représente le nombre de jours depuis le début d'exécution. Si le nombre de jours atteint la valeur prédéfinie, l'horloge centrale bascule de l'état *FinJour* à l'état *FinExécution*

qui marque la fin de l'exécution. Sinon, elle bascule à l'état *DébutJour* pour continuer l'exécution.

Dans le formalisme de l'automate temporisé, si un état ne contient pas d'invariant, l'exécution d'automate peut être bloquée dans cet état. Comme l'horloge avance un pas de temps par jour, les états dans l'horloge centrale sont représentés par les états urgents pour forcer l'exécution d'automate.

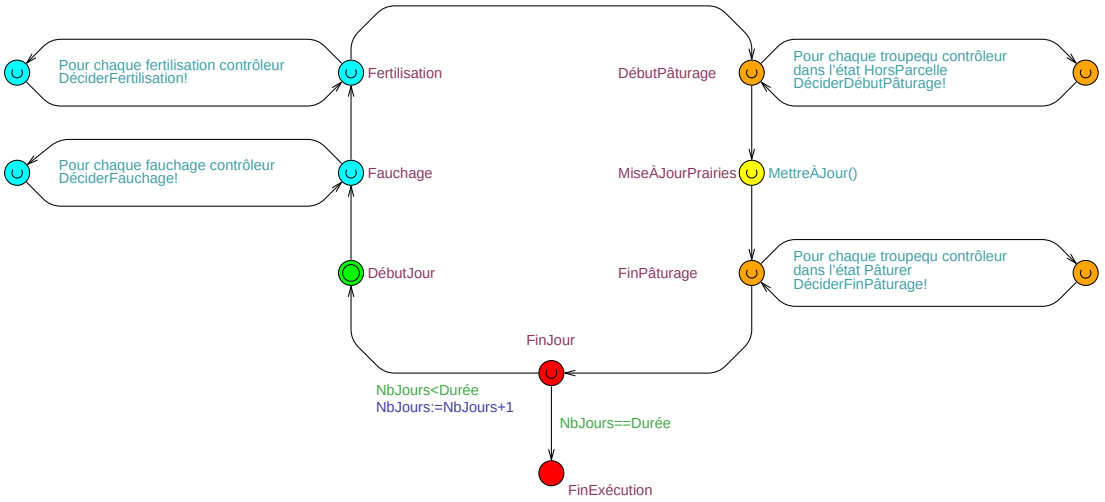


FIG. 5.8 – Automate de l'horloge centrale

5.3.5 Modèle d'exploitation

Les automates de la couche exécution et de la couche contrôleur sont des méta-modèles. Ils doivent être paramétrés pour être utilisés. Étant donné une exploitation de pâturage avec p parcelles et t troupeaux, le modèle de l'exploitation est construite de la façon suivante :

- p variables de hauteur de l'herbe pour p parcelles indexées de 0 à $p-1$ dans un tableau ;
- p automates de fauchage et p automates de fertilisation numérotés avec les valeurs entre 0 et $p-1$ qui correspondent à la numérotation des parcelles ;
- p automates contrôleur de fauchage et p automates contrôleur de fertilisation numérotés avec les valeurs entre 0 et $p-1$ pour pouvoir contrôler les automates ayant la même numérotation dans la couche d'exécution ;
- t automates de mise au pâturage et t automates de contrôleur de mise au pâturage numérotés avec les valeurs entre 0 et $t-1$ pour établir la correspondance entre eux ;
- une horloge centrale.

5.4 Exécution du modèle

Un *model-checker* (voir la section 3.2) est utilisé pour exécuter le modèle d'automates. Les états d'erreur dans les automates de la couche d'exécution sont utilisés pour interrompre l'avancement d'exécution lorsque un automate d'exécution détecte une violation des règles de pâturage. Ces interruptions empêchent l'horloge centrale d'arriver à l'état *FinExécution*. Comme les automates de la couche d'exécutions sont contrôlés par les automates de contrôleur, si un automate d'exécution détecte une violation de règles, c'est forcément l'automate contrôleur qui lui a envoyé une commande inadéquate. Comme les automates de contrôleur modélisent les stratégies des activités de pâturage, la source de violation est sûrement les stratégies. Nous pouvons donc déterminer si les stratégies implémentées dans un modèle sont valides en testant si l'exécution du modèle peut terminée sans être interrompue.

Une façon de tester si une exécution peut arriver à la fin est de poser au *model-checker* la question "L'horloge centrale arrivera-t-elle à l'état *FinExécution* ?" L'expression de cette question en langage TCTL est $\exists\Diamond HorlogeCentrale.FinExcution$. Si la réponse est oui, aucune violation de règle n'est détectée pendant l'exécution et le *model-checker* fournit une trace d'exécution à partir de laquelle des informations utiles peuvent être extraites. Si la réponse est non, un des automates d'exécutions a détecté un problème de violation et aucune trace n'est fournie dans ce cas. D'un certain point de vue, nous pouvons considérer que l'exécution du modèle d'automate par le *model-checker* est une simulation de pâturage car la trace d'exécution fournit les informations complètes sur les activités de pâturage pendant la période d'exécution du modèle.

Uppaal est utilisé comme *model-checker* pour exécuter le modèle de pâturage en automates temporisés. Si l'exécution peut être terminée sans être interrompue par les états d'erreur, c'est-à-dire que les stratégies dans le modèle n'engendrent pas d'action qui viole les règles de pâturage, Uppaal retourne une trace d'exécution à partir de laquelle des informations sont extraites. Une stratégie peut être directement évaluée par ces informations ou par le résultat d'une fonction de coût qui prend ces informations comme paramètres. Voici une liste d'indicateurs qui sont directement extraits de la trace d'exécution :

- le nombre de jours où les troupeaux ont pâturé sur une parcelle ;
- le nombre de jours où les troupeaux ne se sont nourris que d'herbe ;
- la quantité d'herbe totale ingérée par les animaux (kg) ;
- le montant de fourrage conservé et de concentré proposé aux animaux (kg) ;
- la quantité d'herbe totale fauchée (kg) ;
- la quantité d'engrais appliquée sur les parcelles (kg).

Voici un indicateur qui est calculé à partir d'informations extraites :

- Le risque de lixiviation dû aux activités de pâturage (UGP JPE) ($jour{*}animaux/ha$) déterminé avec la formule : $Stress = \frac{JourPâturage{*}NBAnimaux}{Surface}$ dont le nombre de jours de pâturage est calculé de la façon suivante :
  - Chaque jour où les animaux ne se sont nourris que d'herbe cumule 20 heures.
  - Chaque jour où les animaux se sont nourris d'herbe et de la quantité de fourrage *FourrageNiveau1* cumule 20 heures.

– Chaque jour où les animaux se sont nourris d'herbe et de la quantité de fourrage *FourrageNiveau2* cumule 8 heures.

La somme des heures cumulées est divisée par 24 pour avoir le nombre de jours de pâturage.

5.5 Logiciel prototype PaturMata

Le logiciel PaturMata permet de manière automatisée la création du modèle en instanciant les méta-modèles d'automate, le lancement de model-checker avec une requête en langage TCTL et l'extraction des informations à partir de la trace d'exécution. Ce prototype propose une IHM graphique qui permet facilement aux utilisateurs de remplir les descriptions d'exploitation de pâturage, de lancer la requête de validation de stratégies et d'obtenir les résultats des exécutions dans une présentation conviviale.

Onglet Simulation

Cet onglet (figure 5.9) contient les informations générales d'exploitation indépendantes de troupeau et de prairie. Ces informations sont :

– la date du début de simulation ;
– la durée de simulation en nombre de jours ;
– le schéma décadaire de climat.

Onglet Troupeaux

Cet onglet (figure 5.10) permet de gérer les troupeaux dans l'exploitation. On peut ajouter ou supprimer des troupeaux. Pour chaque troupeau ajouté, il faut renseigner le nombre d'animaux.

Onglet Prairie

Les informations sur la configuration de parcelles sont saisies dans cet onglet (figure 5.11). On peut ajouter ou supprimer des parcelles. Les parcelles ont ces attributs suivants :

– la surface de la parcelle (ha) ;
– la densité de l'herbe (kg MS/ha) ;
– le profil décadaire de croissance aussi qu'un ratio permettant de modifier le profil en échelle sans avoir à modifier individuellement toutes les valeurs dans le tableau ;
– la hauteur de l'herbe au premier jour de la simulation (cm) ;
– la distance entre la parcelle et le bâtiment de traite, discrétisée en *Faible*, *Moyenne* et *Grande* ;
– l'hydromorphie du sol ;
– le tableau décadaire de restriction d'accès de la parcelle.

Six types de configuration de parcelle ont été prédéfinis (cf tableau 5.1). Chaque type de configuration indique la surface totale de prairie, le nombre de parcelles, la

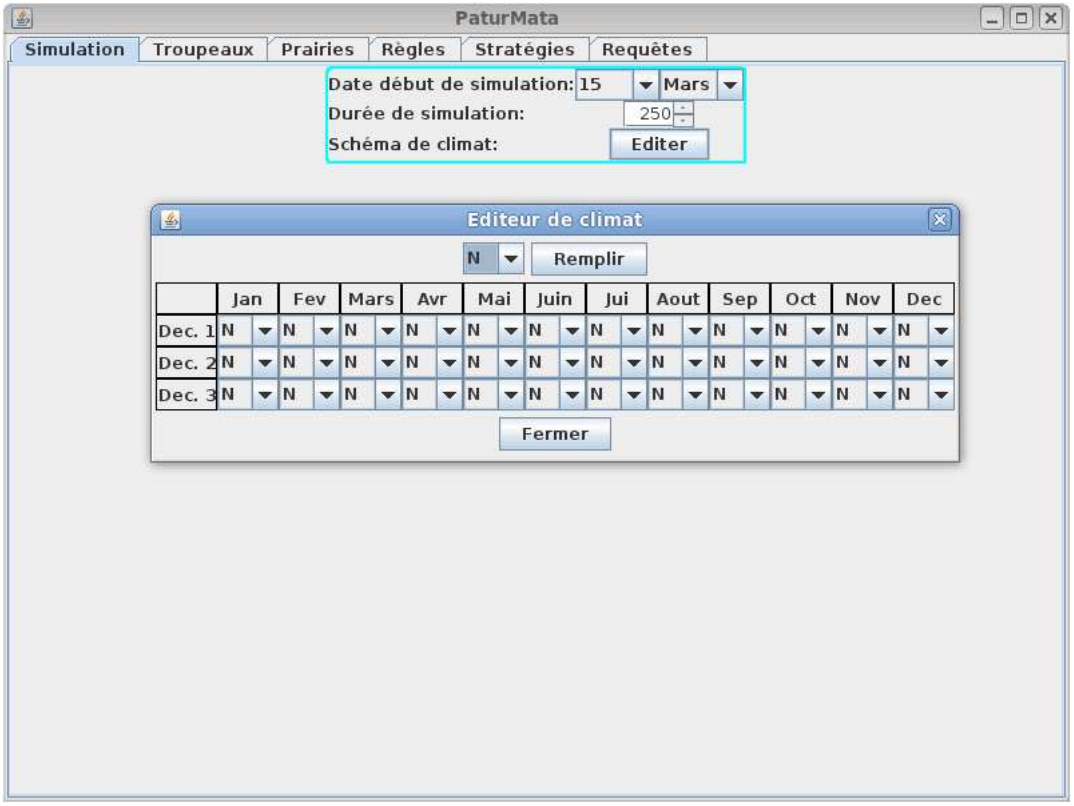


FIG. 5.9 – Onglet Simulation : les informations générales d'exploitation

surface minimum d'une parcelle, la surface maximum d'une parcelle, le pourcentage de surface en sol hydromorphe et la proportion de surface selon les trois catégories de distance. Un générateur implémentant l'algorithme du problème de sac-à-dos [GN72] a été développé qui permet de générer aléatoirement une configuration de parcelles dont les caractéristiques sont conformes au type indiqué.

Onglet Règles

Cet onglet (figure 5.12) affiche les règles de pâturage qui utilisent les paramètres listés dans le tableaux 5.8.

Onglet Stratégies

Un type de stratégies paramétrables pour chaque activité de pâturage a été implémenté dans le logiciel. Ces stratégies sont les plus fréquemment utilisées dans la pratique. L'utilisateur peut modifier les paramètres de ces stratégies sur cet onglet (figure 5.13) sauf que le paramétrage de la stratégie de mise au pâturage est déplacé sur l'onglet *Requête* pour faciliter l'utilisation du logiciel.

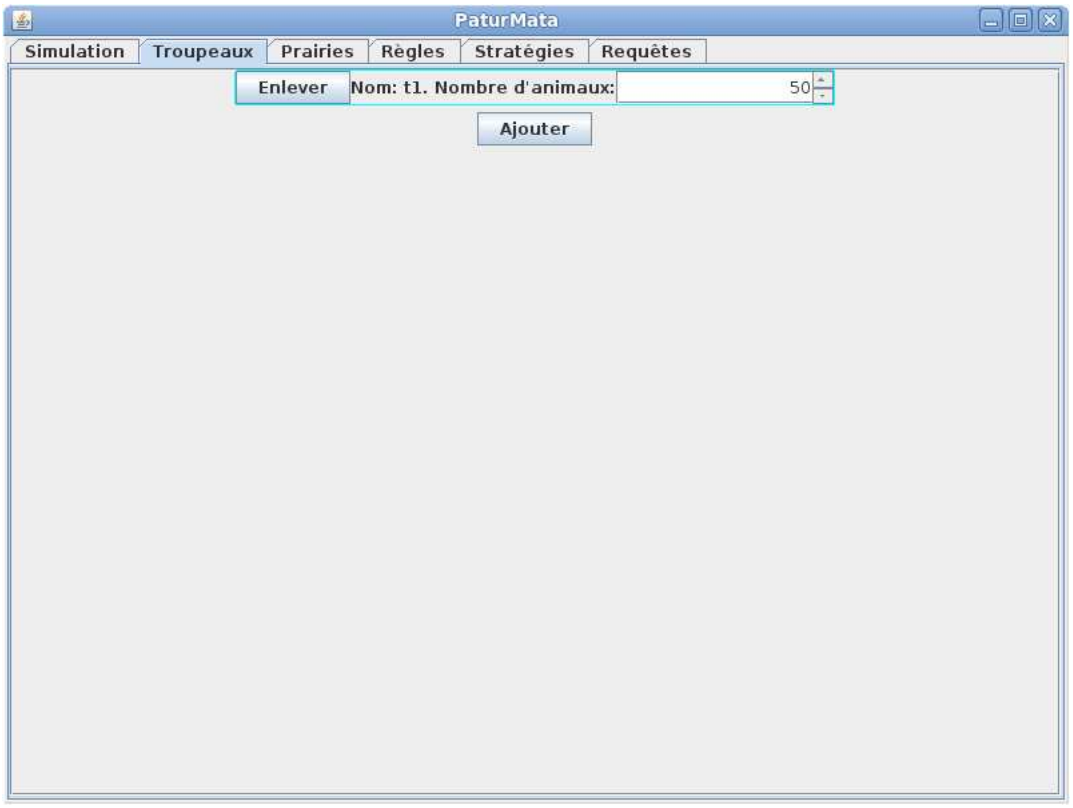


Fig. 5.10 – Onglet Troupeaux : les informations des troupeaux

Les types de stratégies implémentés sont :

– Stratégie de mise au pâturage. La stratégie de mise au pâturage contient trois sous-stratégies :

1. La date de début de la saison de pâturage : C'est la date du démarrage de toutes les activités de pâturage qui peut être différente de la date de début de simulation. La mise au pâturage n'est pas possible avant cette date même si certaines parcelles sont pâturables. Les animaux restent dans le bâtiment de traite et se nourrissent avec du fourrage pendant cette période.
2. Le choix de parcelle et de quantité de fourrage conservé au début de mise au pâturage : Lorsqu'on met un troupeau au pâturage sur une parcelle, deux décisions doivent être prises : 1) la parcelle sur laquelle le troupeau est mis au pâturage, 2) le montant de fourrage conservé offert aux animaux pendant la période où le troupeau reste sur cette parcelle. Pour faire ces deux choix, il faut calculer la quantité d'herbe disponible sur toutes les parcelles où le pâturage est possible (parcelle accessible et dont la hauteur de l'herbe est comprise entre *HauteurPaturageMin* et *HauteurPaturageMax*). On calcule ensuite le nombre de jours prévisionnel de pâturage p en divisant la quantité

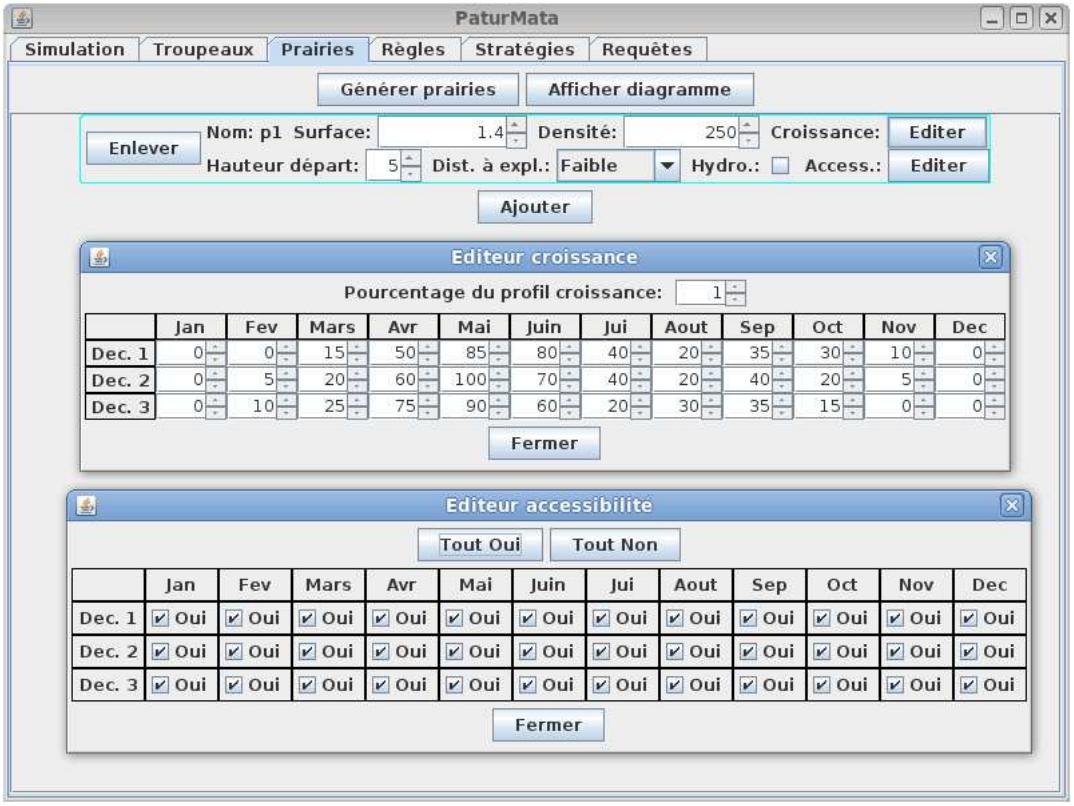


FIG. 5.11 – Onglet Prairie : les informations des parcelles

d'herbe disponible par le besoin journalier d'herbe du troupeau. En fonction de la valeur du p et d'un paramètre t, nombre de jours de pâturage prévisionnel souhaité, les choix sont faits en appliquant les règles suivantes :

- Si $p = 0$ qui signifie qu'aucune parcelle n'est pâturable, le troupeau est laissé dans le bâtiment de traite et nourri entièrement avec du fourrage ;
- Si $0 < p \leq t$ qui signifie que la quantité d'herbe est insuffisante, la parcelle pâturable sur laquelle l'herbe est la plus haute est choisie. Pendant la durée de pâturage sur cette parcelle, un montant de fourrage conservé *FourConsNiveau2* est offert aux animaux tous les jours.
- Si $t < p \leq 2t$ qui signifie que la quantité d'herbe est modérée, parmi toutes les parcelles pâturables sur lesquelles la hauteur de l'herbe est comprise entre *HauteurPaturageMin* et *HauteurPaturageMaxAlt*, la parcelle sur laquelle l'herbe est la plus haute est choisie. Pendant la durée de pâturage sur cette parcelle, un montant du fourrage conservé *FourConsNiveau1* est offert aux animaux tous les jours. La hauteur *HauteurPaturageMaxAlt* est légèrement plus petite que *HauteurPaturageMax*, ce qui permet aux animaux de se nourrir plus facilement.

Type configuration	Surface totale (ha)	Nombre parcelles	Surface min (ha)	Surface max (ha)	% surface hydro-morphe	Proportion surface distance
Petite 1	12,5	7	1,0	4,0	0	1 : 1 : 0
Moyenne 1	22,5	12	1,0	4,0	0	1 : 1 : 1
Grande 1	35,0	20	1,0	4,0	15	1 : 1 : 1
Petite 2	12,5	7	1,0	4,0	50	1 : 1 : 0
Moyenne 2	22,5	12	1,0	4,0	50	1 : 1 : 1
Grande 2	35,0	20	1,0	4,0	50	1 : 1 : 1

Tab. 5.1 – Types prédéfinis de configuration de parcelles

  - Si $2t < p$ qui signifie que la quantité d'herbe est suffisante, on fait le même choix de parcelle que dans le cas du stock modéré et aucun fourrage conservé n'est offert au troupeau pendant le pâturage.

  Dans tous les cas, si la hauteur de l'herbe sur plusieurs parcelles est égale (après avoir arrondi au centimètre), la parcelle la plus proche du bâtiment de traite est choisie.

3. La sortie d'un troupeau d'une parcelle à la fin de mise au pâturage : Pour profiter au maximum de l'herbe sur la prairie, les animaux sont laissés sur une parcelle aussi longtemps possible sans que la hauteur de l'herbe soit inférieure à *HauteurFinPâturage*.

– Stratégie de fauchage. La stratégie de fauchage implémentée est de faucher une parcelle dès que l'herbe est trop haute ($>$*HauteurPaturageMax*) pour mettre le troupeau.
– Stratégie de fertilisation. La stratégie de fertilisation implémentée simule une application systématique d'une dose d'engrais sur une parcelle *DélaiFertilisation* jours après le départ de troupeau ou un fauchage, en respectant le quota annuel d'engrais sur cette parcelle. L'utilisateur peut activer cette stratégie de fertilisation ou interdire complètement la fertilisation sur le terrain.

Onglet Requête

L'utilisateur lance des simulations sur cet onglet. De plus, le choix du paramètre t, nombre de jours prévisionnels de pâturage souhaité, de la stratégie de mise au pâturage peut être fait sur cet onglet. Pour pouvoir comparer plus facilement les résultats de simulation pour différentes valeurs du paramètres t, l'utilisateur peut choisir un intervalle de valeurs pour t et les résultats de simulation sont affichés ensemble.

La figure 5.14 montre l'affichage du résultat d'une simulation avec le paramètre $t = 7$ pour la stratégie de mise au pâturage, avec l'activation de fertilisation. La colonne JP (nombre de jours de pâturage) montre que le troupeau a pâturé pendant 204 jours au total pendant la durée de simulation. La colonne JPS (nombre de jours de pâturage sans fourrage conservé) montre que le troupeau a pâturé pendant 58 jours sans avoir

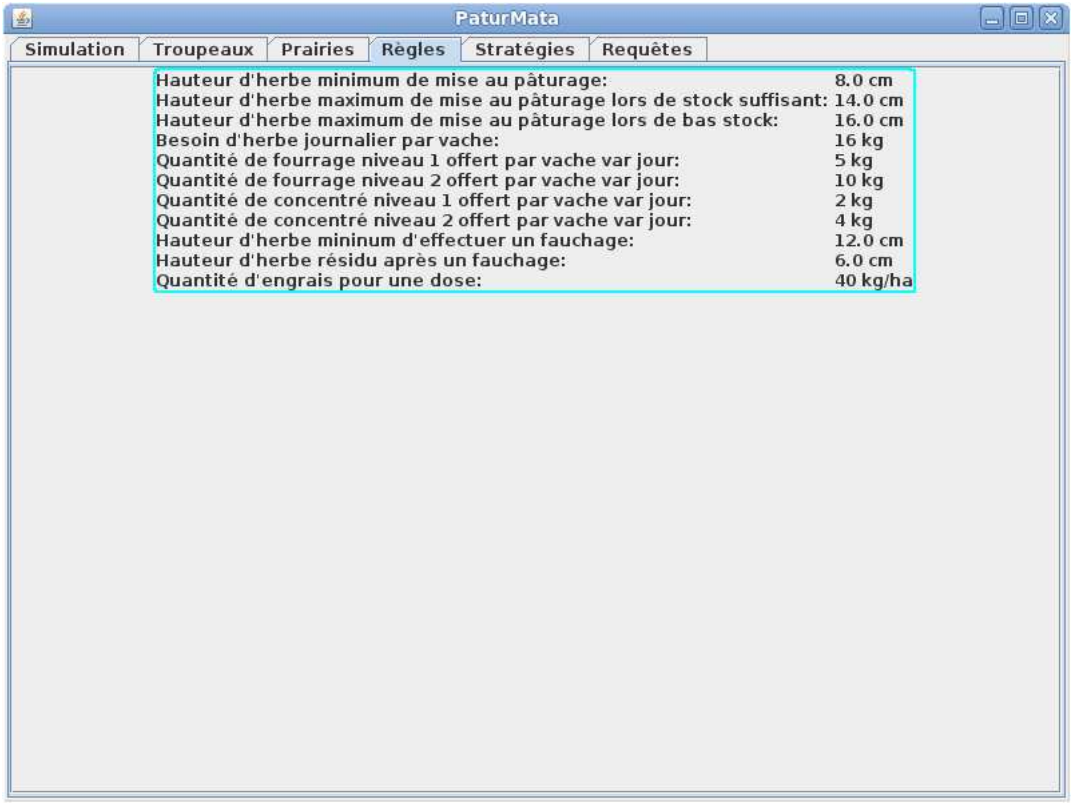


FIG. 5.12 – Onglet Règles : l'affichage des règles

de fourrage comme aliment supplémentaire. La colonne CH (chargement) montre que le stress du sol du à l'activité de pâturage est de 327 jours animaux/ha. La colonne HE (herbe) indique que la quantité d'herbe ingérée directement de la prairie est de 63328 kg. La colonne FO (fourrage conservé) indique que le montant de fourrage proposé aux animaux est de 56650 kg. La colonne FA (fauchage) indique que la quantité d'herbe fauchée est de 0 kg. Finalement le colonne FE (fertilisation) indique qu'aucune fertilisation n'a été appliquée pendant la simulation.

Exemple

Voici un exemple d'utilisation de PATURMATA sur l'exploitation suivante : 1 troupeau de 50 vaches laitières, une prairie de 35 ha divisée en 20 parcelles dont la surface est entre 1 et 4 ha. Les simulations commencent le 16 mars pour une durée de 200 jours. Six simulations ont été lancées avec six valeurs de t (nombre de jours de pâturage prévisionnel souhaité). Le tableau 5.2 montre une comparaison des indicateurs les plus intéressants des résultats de simulation.

Un des critères souvent utilisés pour évaluer les scénarios de pâturage est de comparer la quantité de fourrage conservé utilisé et la quantité d'herbe fauchée. Le fourrage

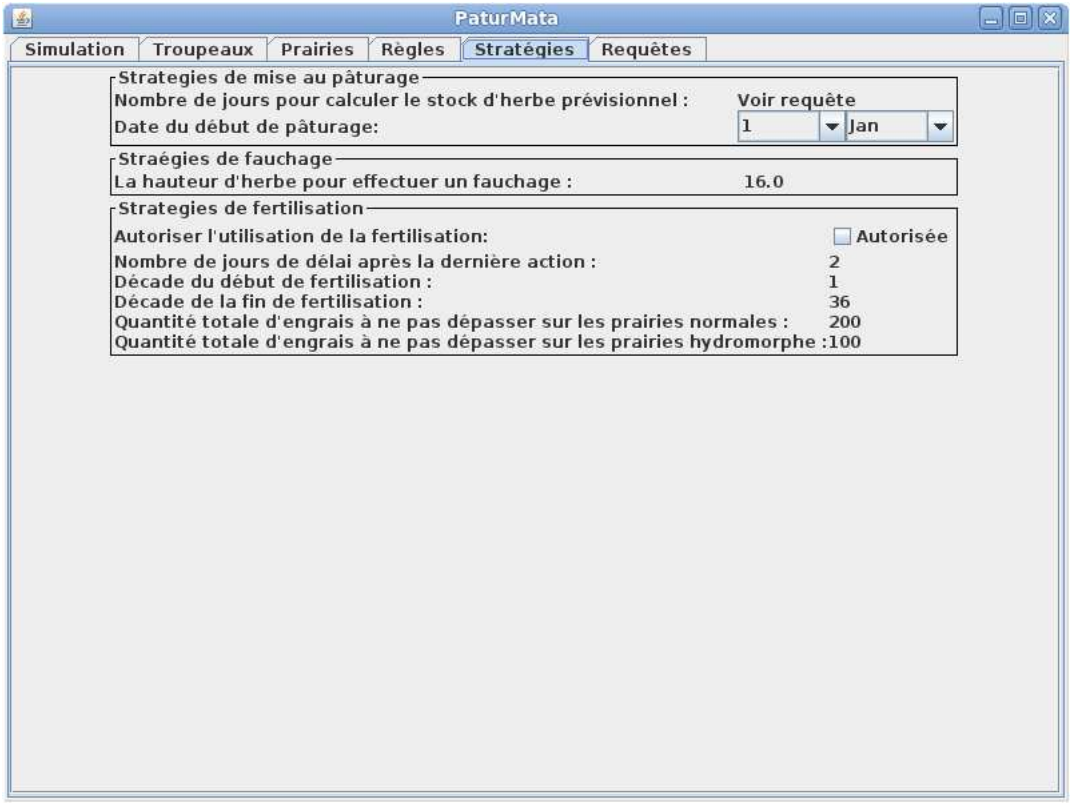


FIG. 5.13 – Onglet Stratégies : les stratégies de fauchage et de fertilisation

conservé peut être produit à partir de l'herbe fauchée ou être simplement acheté. Le premier demande de la main-d'œuvre alors que le dernier impose une dépense supplémentaire. Le fauchage d'une parcelle demande également de la main-d'œuvre et une dépense d'essence pour la machine de fauchage. On cherche donc le scénario de pâturage avec les valeurs les plus petites possibles pour la quantité de fourrage conservé consommé et d'herbe fauchée. Parmi les six exécutions, celle avec $t = 6$ est le scénario optimal car la quantité de fourrage conservé offert aux animaux et la quantité d'herbe fauchée sont minimums. On peut conclure que $t = 6$ est la meilleur valeur du paramètre de stratégie de mise au pâturage.

5.6 Validation du modèle

Le modèle PATURMATA a été implémenté en langage UPPAAL. Les fonctions de calcul d'herbe ont été implémentées en langage C. Le modèle PATURMATA s'inspire du modèle numérique PATUR'IN. Les difficultés techniques que j'ai rencontrées ont été les suivantes :

– Faute de type flottant de variable (float, double) dans UPPAAL, les variables de

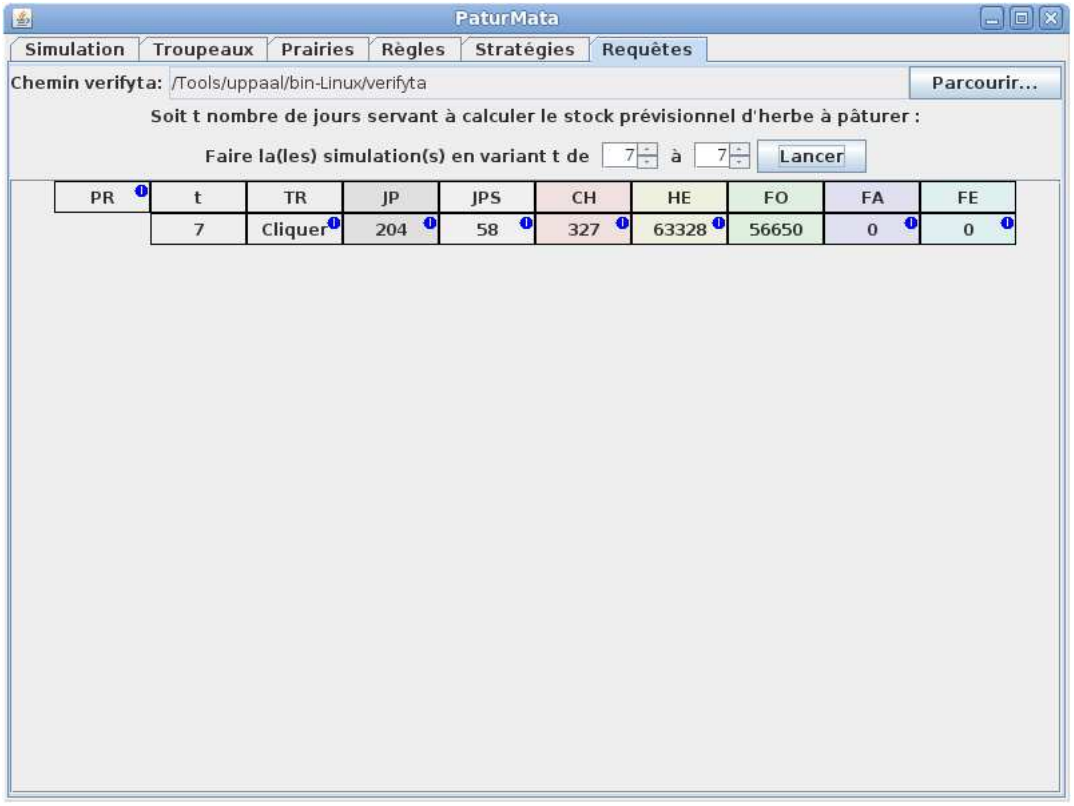


FIG. 5.14 – Onglet Requête : le paramétrage de la stratégie de mise au pâturage et le lanceur de requêtes

hauteur de l'herbe sont de type entier. Afin de conserver au maximum la précision, la hauteur de l'herbe est représentée par les valeurs 1000 fois plus grandes que la hauteur réelle (1000 représente 1 centimètre).

– Faute de librairie de calcul de logarithme qui est indispensable pour calculer la dynamique de l'herbe, j'ai conçu une fonction approximative du logarithme.

Les deux solutions listées ci-dessus introduisent inévitablement des erreurs de calcul. Il a donc été nécessaire de valider le modèle numérique implémenté dans PATURMATA et de déterminer la nécessité de calibration du modèle.

1. Croissance de l'herbe et consommation de l'herbe. La validation a été faite en comparant les résultats de simulation. Comme le modèle dans PATURMATA est une version dérivée du modèle de PATUR'IN, une version spéciale de PATUR'IN qui prend en compte les restrictions supplémentaires et les simplifications (cf section 5.3) a été conçue pour pouvoir faire la comparaison. Nous avons généré aléatoirement une exploitation sur une prairie de 35ha divisée en 10 parcelles et un troupeau de 50 vaches. La même séquence d'actions a été appliquée dans PATUR'IN et PATURMATA. La hauteur de l'herbe dans la trace de simulation a été

Jours prévisionnel (t)	4	6	8	10	12	14
Jours pâturage sans fourrage conservé offert	142	147	132	107	72	66
Consommation d'herbe (kg)	143500	144493	139740	132987	127991	126494
Fourrage conservé offert (kg)	16500	15500	20250	27000	32000	33500
Herbe fauchée (kg)	28129	24906	26595	30774	36756	36636

TAB. 5.2 – Comparaison partielle des résultats d'exécutions avec six valeurs de t

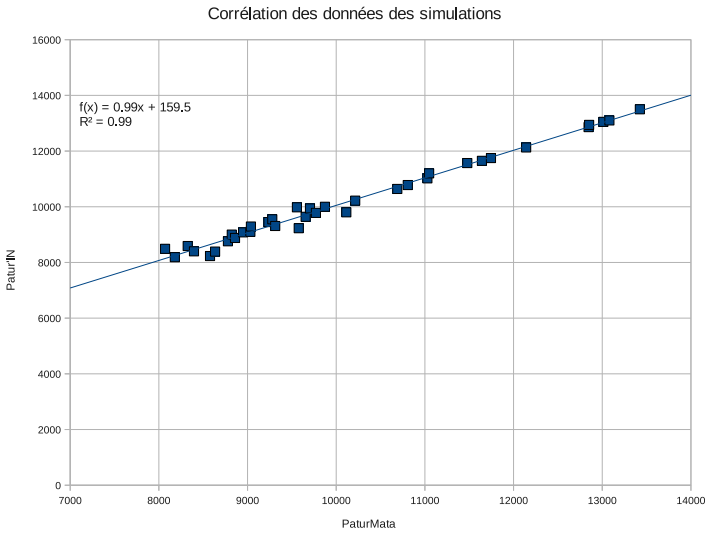


FIG. 5.15 – Corrélation de la hauteur de l'herbe lors de l'entrée d'un troupeau dans une parcelle

comparée pour déterminer si les fonctions de calcul de la dynamique de l'herbe étaient cohérentes. La comparaison a porté sur la hauteur de l'herbe le jour où le troupeau entre dans une parcelle, et le jour où le troupeau sort d'une parcelle, la hauteur d'entrée étant le résultat du calcul de croissance des jours précédents et la hauteur de sortie étant le résultat du calcul de croissance dégradée et de consommation pendant le pâturage. Une analyse de corrélation montre que les valeurs de hauteur d'entrée sont différentes de 0.27 centimètres alors que les valeurs de hauteur de sortie sont différentes de 0.64 centimètres. À cause de ces biais, les simulations sont assez différentes au bout de 200 jours de simulation. Par exemple la hauteur de l'herbe dans une parcelle dans PATURMATA est supérieure à *HauteurPâturageMin* tandis que la hauteur de l'herbe dans la même parcelle dans PATUR'IN est inférieure. Il n'est donc pas possible d'appliquer la même action. Il a donc été nécessaire de calibrer le modèle dans PATURMATA pour que la simula-

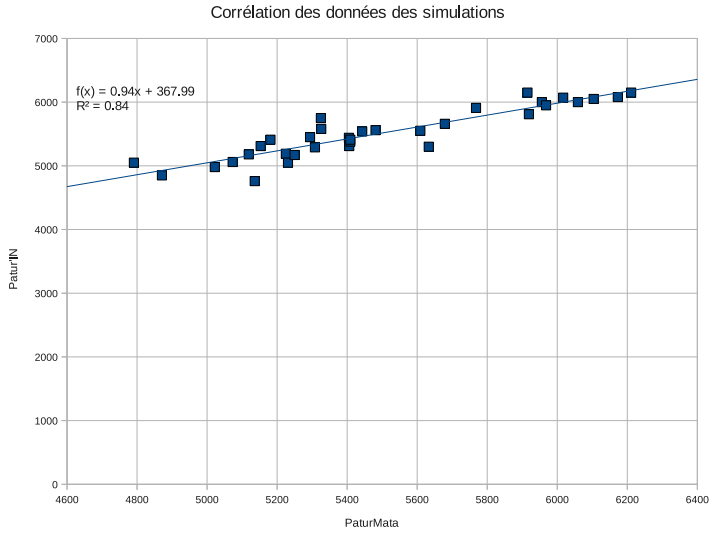


FIG. 5.16 – Corrélation de la hauteur de l'herbe lors de la sortie d'un troupeau d'une parcelle

tion pendant une saison de pâturage reste cohérente. La calibration a été réalisée par l'ajout d'une constante dans le résultat du calcul de hauteur de l'herbe afin de diminuer le biais. Une deuxième comparaison a montré que les valeurs étaient suffisamment corrélées, $h_{pin} = 0,99 * h_{pm} + 159,5$ pour les valeurs de hauteur d'entrée et $h_{pin} = 0,94 * h_{pm} + 367,99$ pour les valeurs de hauteur de sortie où h_{pin} est la hauteur dans PATUR'IN et h_{pm} est la hauteur dans PATURMATA (figures 5.15 et 5.16).

2. Effet de fertilisation. L'effet de fertilisation a été validé de façon statistique sans l'aide de PATUR'IN. Sur la même configuration de parcelles et de troupeau, des simulations ont été lancées avec la même stratégie de pâturage mais différentes stratégies de fertilisation. À l'aide des statistiques fournies à la fin des simulations, l'efficacité de fertilisation peut être calculée à l'aide de la formule suivante :

$$Efficacité = \frac{RendementHerbeFertilisé - RendementHerbe}{QuantitéEngraisAppliqué} \tag{5.1}$$

Les simulations suivantes ont été lancées sur 3 types d'exploitation.
– Config 1 : 50 vaches, 35ha de prairie divisée en 10 parcelles soit 70 ares/vache.
– Config 2 : 50 vaches, 20ha de prairie divisée en 10 parcelles soit 40 ares/vache.
– Config 3 : 50 vaches, 12,5ha de prairie divisée en 10 parcelles soit 25 ares/vache.
Comme montré dans le tableau 5.3, l'indicateur de l'efficacité de fertilisation à la sortie de notre simulateur est compris dans l'intervalle [13, 19]. En se basant sur l'avis de l'expert, le calcul de l'effet de fertilisation est considéré satisfaisant.

Config	Herbe ingérée (kg MS)	Fourrage ingéré (kg MS)	Herbe fauchée (kg MS)	Engrais (kg N/ha)	Rendement (kg MS/ha)	Efficacité fertilisation
Config 1	175752	24250	16445	0	5491	-
	162034	13000	28775	0	5452	-
	200023	0	116547	192	9045	19
	175028	0	141385	198	9040	18
Config 2	106892	93100	0	0	5345	-
	107587	68100	0	0	5379	-
	148497	21500	37931	255	9321	16
	138740	36250	46987	247	9286	16
Config 3	65797	134200	0	0	5264	-
	67497	108000	0	0	5400	-
	107497	92500	2531	246	8802	14
	97093	78600	10711	227	8624	14

TAB. 5.3 – Validation du modèle par test de l'efficacité de la fertilisation

5.7 Application aux données du bassin versant du Yar

La pollution par le nitrate à cause des activités d'élevage est devenue sévère dans certaines régions. Le bassin versant du Yar a fait l'objet de nombreux travaux sur le lien entre élevage bovin laitier et émission d'azote [Mfdrpldlm99]. Nous avons choisi des données de pâturage de ce bassin versant comme cas d'application de notre modèle PATURMATA du fait de la disponibilité des données.

Les données de pâturage contiennent la description d'exploitation et de leurs statistiques de pâturage. Le tableau 5.4 montre les données brutes de quatre exploitations. Nous avons ajusté certaines données pour qu'elles correspondent aux spécifications du modèle PATURMATA.

1. La surface minimale d'une parcelle est très petite (par exemple 0,21 ha pour le type 1). Dans la réalité, les petites parcelles voisines sont souvent utilisées comme une grande parcelle pour faire un pâturage tournant. Nous avons réduit le nombre de parcelles pour que la surface minimum de chaque parcelle soit environ 1 ha.
2. Le nombre d'animaux compte les génisses qui ne consomment pas autant d'herbe que les adultes alors que le modèle PATURMATA demande le nombre de vache (UGB). Nous avons converti le nombre d'animaux en unité UGB.

La description des exploitation après ces modification sont montrées dans le tableaux 5.5. Les données de pâturage du bassin versant du Yar sont sous forme de statistiques de scénario de pâturage. Elles contiennent les indicateurs suivants :

- NP : nombre moyen de cycle de pâturage par parcelle (entre 3 et 5 par an)
- NF : nombre moyen de fauches par parcelle (inférieur à 2 par an)
- TP : nombre moyen de parcelles dédiées au pâturage (entre 25% et 50% du nombre total de parcelles)
- TF : nombre moyen de parcelles dédiées au fauchage (inférieur à 3)

- TM : nombre moyen de parcelles pour les pratiques mixtes (entre 50% et 75% du nombre total de parcelles)
- PP : durée moyenne entre deux pâturages (entre 30 et 40 jours)
- FF : durée moyenne entre deux fauchages (supérieur à 50 jours)
- PF : durée moyenne entre un pâturage et un fauchage (entre 40 et 50 jours)
- FP : durée moyenne entre un fauchage et un pâturage (entre 40 et 50 jours)

Dix exploitations ont été générées pour chaque type d'exploitation. À partir des traces d'exécution, nous avons calculé les indicateurs du tableau 5.6. Les résultats de simulations sont proches des données observées confirmant la pertinence du modèle PaturMata.

	Surface (ha)	NB parcs	Surface parcs	Nombre d'animaux (laitière)	% surface dist. proche	% surface dist. moyenne	% surface dist. loin	Fertilisation (kgN/ha/an)
Type 1	70	28	[0,21 9,61]	90 (90)	59,18	24,49	16,33	[120 180]
Type 2	70	34	[0,25 8,5]	90 (55)	34,88	37,21	27,91	0
Type 3	95	52	[0,18 6,21]	180 (71)	43,28	28,36	28,36	0
Type 4	55	40	[0,24 6,45]	180 (60)	41,18	47,06	11,76	[120 180]

TAB. 5.4 – Données de pâturage du bassin versant de Yar

	Surf. (ha)	NB parcs	Surface parcs	UGB	% surface dist. proche	% surface dist. moyenne	% surface dist. loin	Fertilisation (kgN/ha/an)
Type 1	70	24	[1,0 9,61]	90	59,18	24,49	16,33	[120 180]
Type 2	70	28	[1,0 8,5]	90	34,88	37,21	27,91	0
Type 3	95	30	[1,0 6,21]	140	43,28	28,36	28,36	0
Type 4	55	30	[1,0 6,45]	90	41,18	47,06	11,76	[120 180]

TAB. 5.5 – Données ajustées de pâturage du bassin versant de Yar

	NP	NF	TP	TF	TM	PP	FF	PF	FP
Type 1	3	1	7	4	13	47	140	-	42
Type 2	3	1	19	0	9	54	-	113	56
Type 3	3	1	13	0	11	49	-	120	55
Type 4	4	1	11	0	19	42	-	75	44

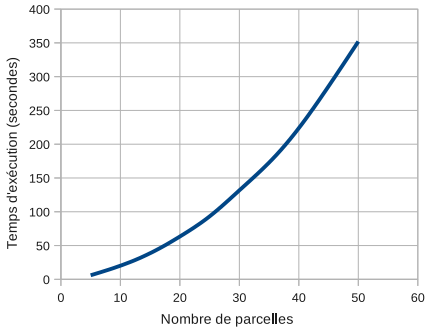
TAB. 5.6 – Valeurs des indicateurs des simulations de pâturage du bassin versant de Yar

5.8 Conclusion

Ce chapitre a présenté PaturMata, une approche de modélisation hybride de la gestion de pâturage basé sur des automates temporisés. Cette modélisation hybride permet de se focaliser principalement sur l'es interactions entre la dynamique de l'herbe dans la prairie et les activités de pâturage : la mise au pâturage, le fauchage et la fertilisation. Elle permet aussi d'implémenter des stratégies de pâturage. Le *model-checking* est utilisé pour exécuter le modèle. Une trace d'exécution est fournie à la fin de l'exécution à partir de laquelle des informations utiles peuvent être extraites pour calculer les indicateurs permettant d'évaluer les stratégies implémentées.

Afin de mieux séparer les rôles dans le système, les automates sont organisés dans une hiérarchie composée de quatre couches : la couche prairie (modèle biologique), la couche d'exécution (modèle d'activité de pâturage), la couche contrôleur (modèle décisionnel) et l'horloge centrale (modèle du temps). La couche prairie a été conservée numérique dans le modèle qualitatif pour préserver au maximum la précision du calcul d'herbe qui s'inspire du modèle numérique de pâturage Patur'IN. Cette particularité est pertinente au niveau du calcul. Cependant, l'existence d'une partie numérique dans un modèle en automates temporisés ne permet pas à l'algorithme du *model-checking* d'atteindre sa performance maximum pendant l'exécution. Cette approche hybride a été gardée car le temps d'exécution reste acceptable pour une utilisation interactive. Le tableau 5.7 montre le temps d'exécution en fonction du nombre de parcelles dans l'exploitation. Toutes les exploitations concernent un seul troupeau de 50 animaux, commencent le 1er mars pour une durée de 250 jours.

Le modèle PaturMata a servi de base pour la synthèse de stratégies de pâturage qui sera l'objet du chapitre suivant.



Nb parcelles	5	10	15	20	25	30	35	40	45	50
Tps d'exécution (s)	6	19	37	63	91	132	171	221	285	352

Tab. 5.7 – Temps de exécutions en fonction de nombre de parcelles. Toutes les exploitations contiennent un troupeau de 50 animaux. Toutes les exécutions commencent le 1er mars pour une durée de 250 jours. Test réalisées sur une machine de Xeon 3GHz, 6 Go mémoire vive, Fedora 12 x86_64, Java SE 7.

Paramètre	Description	Valeur
HauteurPâturageMin	Hauteur de l'herbe minimum pour mettre un troupeau au pâturage	8 cm
HauteurPâturageMax	Hauteur de l'herbe maximum pour mettre un troupeau au pâturage	16 cm
HauteurPâturageMaxAlt	Hauteur de l'herbe maximum pour mettre un troupeau au pâturage lors de stock d'herbe suffisant	14 cm
HauteurFinPâturage	Hauteur de l'herbe minimum pour terminer un pâturage	Calculé pendant l'exécution
HauteurAprèsFauchage	Hauteur de l'herbe après un fauchage	6 cm
HauteurMinFauchage	Hauteur de l'herbe minimum pour déclencher un fauchage	16 cm
FourConsNiveau0	Quantité de fourrage conservé proposée aux animaux niveau 0	0 kg/animal/jour
FourConsNiveau1	Quantité de fourrage conservé proposée aux animaux niveau 1	5 kg/animal/jour
FourConsNiveau2	Quantité de fourrage conservé proposée aux animaux niveau 2	10 kg/animal/jour
ConcentréNiveau0	Quantité de concentré proposée aux animaux niveau 0	0 kg/animal/jour
ConcentréNiveau1	Quantité de concentré proposée aux animaux niveau 1	2 kg/animal/jour
ConcentréNiveau2	Quantité de concentré proposée aux animaux niveau 2	4 kg/animal/jour
FD1	Période avant que la fertilisation soit effective	3 jours
FD2	Période pendant laquelle la fertilisation est effective	18 jours
FD3	Période pendant laquelle la fertilisation est effective mais interruptible par le pâturage ou le fauchage	39 jours
DoseEngrais	Dose d'engrais d'une application	40 kg N/ha
QuotaEngrais	Quota d'engrais pendant un an sur une parcelle normale	200 kg N/ha/an
QuotaEngraisHydro	Quota d'engrais pendant un an sur une parcelle hydromorphe	100 kg N/ha/an
DélaiFertilisation	Délai entre la fin de pâturage et une application d'engrais	2 jours

Tab. 5.8 – Paramètres du modèle de pâturage

Chapitre 6

Recherche de stratégies de gestion de pâturage

6.1 Introduction

Le modèle PaturMata présenté dans le chapitre 5 permet de comparer les modes de gestion du pâturage selon les différentes stratégies. Il est clair que l'application des différentes stratégies entraîne différents scénarios. Les stratégies sont évaluées à l'aide d'une fonction d'évaluation à partir des informations contenues dans les scénarios. Cependant, la recherche de la stratégie optimale qui maximise ou minimise une fonction reste difficile étant donnée les nombreuses possibilités. Nous présentons, dans ce chapitre, les méthodes de recherche de stratégies optimales développées en prolongement de ce qui a été présenté sur EcoMata au chapitre 4.

Parmi les trois activités de pâturage modélisés dans PaturMata, la mise au pâturage, le fauchage et la fertilisation, l'activité de fauchage suit des règles simples (voir la section 5.3.2.2) tenant compte de la pratique effective et ne nécessite pas de recherche de stratégie optimale. Nous nous focalisons sur la recherche de stratégies de mise au pâturage et de stratégies de fertilisation. La section 6.2 présente la recherche de stratégies de mise au pâturage. La section 6.3 introduit la définition de l'automate de coût qui sert au support de la recherche de stratégies de fertilisation. Les sections 6.4 présentent la recherche de stratégies de fertilisation datées en utilisant la synthèse de contrôleur. La section 6.5 et 6.6 présentent deux méthodes de génération de stratégies génériques de pâturage combinant synthèse de contrôleur et apprentissage supervisé. Le chapitre se termine par une conclusion dans la section 6.7.

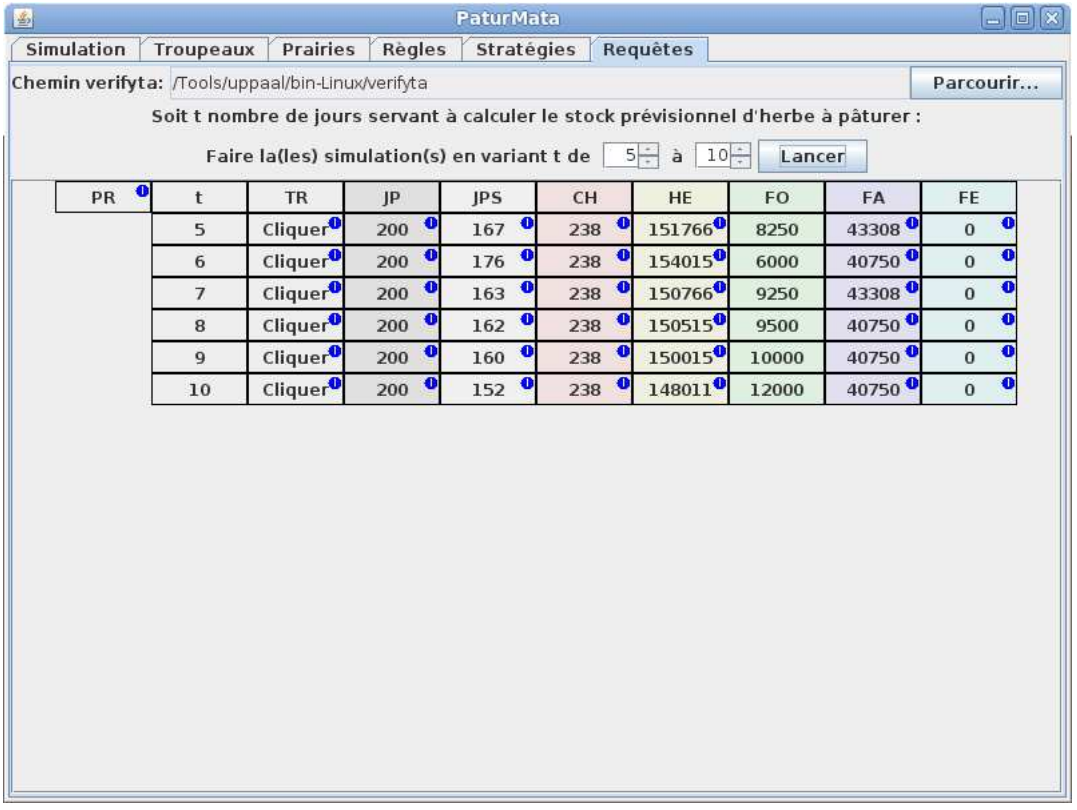
6.2 Synthèse de stratégie de mise au pâturage

Comme décrit dans la section 5.5, la stratégie de mise au pâturage est composée de deux sous-stratégies, chacune paramétrable :

– La date de début de pâturage ;

– Le nombre de jours prévisionnel de pâturage t qui détermine la quantité de fourrage conservé offert aux animaux pendant la durée de pâturage.

Ces deux paramètres sont caractérisés par des valeurs numériques discrètes. Ceci permet d'utiliser la méthode "générer et tester" présentée dans la section 4.5. Le logiciel PaturMata implémente une fonctionnalité qui permet de lancer automatiquement les exécutions avec différentes valeurs de t dans un intervalle donné. Comme montré dans la figure 6.1, on peut choisir la borne inférieure et la borne supérieure du paramètre t avant de lancer les exécutions. PaturMata affiche les résultats des exécutions que l'on compare afin de choisir la valeur optimale de t.



PR	t	TR	JP	JPS	CH	HE	FO	FA	FE
	5	Cliquer	200	167	238	151766	8250	43308	0
	6	Cliquer	200	176	238	154015	6000	40750	0
	7	Cliquer	200	163	238	150766	9250	43308	0
	8	Cliquer	200	162	238	150515	9500	40750	0
	9	Cliquer	200	160	238	150015	10000	40750	0
	10	Cliquer	200	152	238	148011	12000	40750	0

Fig. 6.1 – Logiciel PaturMata onglet requête : le lancement automatique d'exécutions avec différentes valeurs de t dans un intervalle choisi

Cette méthode "générer et tester" est pertinente pour une utilisation interactive grâce au court temps d'exécution (cf. tableau 5.7) et convient pour la recherche d'une stratégie de mise au pâturage.

Elle a été testée pour la synthèse de stratégie de fertilisation. Mais elle s'est avérée inadéquate pour la raison suivante. Comme présenté dans la section 5.3.2.3, la date de fertilisation possible est restreinte à deux jours après le départ du troupeau ou deux jours après un fauchage. Ces dates ne sont connues qu'au cours de l'exécution et ne peuvent

pas être planifiées. Il a été décidé d'explorer l'approche synthèse de contrôleur sur des automates temporisés de coût afin d'identifier les stratégies optimales, vis à vis d'une fonction de coût donnée de fertilisation. Nous présentons la définition de l'automate temporisé de coût dans la section suivante.

6.3 Automates temporisés de coût

L'automate temporisé de coût est une extension de l'automate temporisé en y associant une fonction de coût. Ce formalisme permet à l'algorithme de *model-checking* d'effectuer la recherche de trajectoire optimale en plus de la vérification d'atteignabilité. Ce formalisme a été choisi pour la représentation du modèle de pâturage. Nous présentons dans cette section la définition formelle de l'automate temporisé de coût et l'outil UPPAAL-CORA qui implémente ce formalisme et permet de trouver l'exécution optimale.

6.3.1 Définition

Un automate temporisé de coût [BLR05] est une extension d'un automate temporisé en y ajoutant une valeur non-négative à chaque état et à chaque transition.

Définition 6.3.1. L'automate temporisé de coût est une extension de l'automate temporisé dénoté par $PTA =< TA, \mathcal{C} >$ où $\mathcal{C} = \mathcal{Q} \cup \mathcal{T} \rightarrow \mathbb{N}$ associe des taux de coût aux états et des coûts aux transitions.

Pendant une exécution (cf section 3.2) d'un automate temporisé de coût :

- si deux configurations sont reliées par un *passage de temps* dénoté par $\langle q_i,\ v_i \rangle \xrightarrow{\Delta\tau_i} \langle q_{i+1},\ v_{i+1} \rangle$, le coût est le taux de coût associé à l'état q_i multiplié par la durée passée dans cet état $\mathcal{C}(q_i) * \Delta\tau_i$;
- si deux configurations sont reliées par une *action discrète* e dénotée par $\langle q_i,\ v_i \rangle \xrightarrow{e} \langle q_{i+1},\ v_{i+1} \rangle$, le coût est $\mathcal{C}(e)$

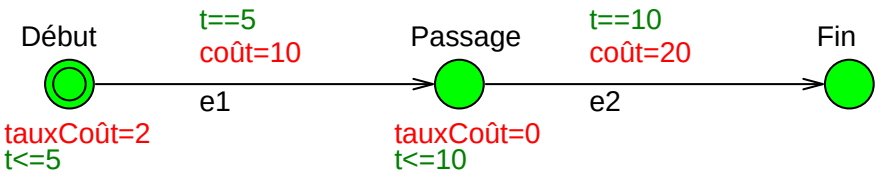


FIG. 6.2 – Un exemple d'automate temporisé de coût

Le coût d'une exécution finie d'un automate temporisé de coût est donc la somme de tous les coûts de la suite d'évolution associées à cette exécution. La figure 6.3.1 montre un exemple d'automate temporisé de coût avec une horloge t. L'exécution de l'automate commence par l'état *Début*. L'invariant de l'état *Début* $t <= 5$ et la garde de la transition sortant de l'état *Début* $t == 5$ indique que l'automate doit rester dans

l'état *Début* exactement pendant 5 unités de temps. Comme l'état *Début* est étiquetée par $tauxCoût = 2$, ce passage de temps augmente le coût total de $2 \times 5 = 10$. L'automate passe, quand l'horloge $t == 5$, de l'état *Début* à l'état Passage grâce à la transition $e1$ étiquetée par $coût = 10$. Cette action discrète augmente instantanément le coût total de 10. L'automate reste ensuit dans l'état *Passage* pendant 5 unités de temps. Mais ce passage de temps n'augmente pas le coût total parce que l'état *Passage* est étiquetée par $coût = 0$. L'automate passe, quand l'horloge $t == 10$, de l'état *Passage* à l'état *Fin* grâce à la transition $e2$ étiquetée par $coût = 20$. Cette action discrète augmente le coût total de 20. Le coût total de l'exécution de l'automate est la somme des coûts qui s'élève à 40.

$$\langle Début,\ 0\rangle \xrightarrow[10]{5} \langle Début,\ 5\rangle \xrightarrow[10]{e1} \langle Passage,\ 5\rangle \xrightarrow[0]{5} \langle Passage,\ 10\rangle \xrightarrow[20]{e2} \langle End,\ 10\rangle$$

6.3.2 Uppaal-Cora

Le logiciel Uppaal-Cora[1] [BLR05] est une extension d'Uppaal qui est spécialement conçue pour effectuer l'analyse d'atteignabilité avec coût optimal sur un réseau d'automates temporisés de coût. Uppaal-Cora prend en entrée un ensemble d'automates temporisés de coût et une requête exprimée dans le langage TCTL. Les automates temporisés de coût dans Uppaal-Cora sont obtenus en ajoutant les éléments suivants dans le langage de description d'automate temporisé dans Uppaal :

- L'expression $cost' == c,\ c \in \mathbb{R}^+$ dans l'invariant d'un état exprime le taux de coût dans cet état.
- L'expression $cost+ = c,\ c \in \mathbb{R}^+$ dans l'étiquette de la mise à jour d'une transition exprime le coût du déclenchement de cette transition.
- Une variable *remaining* peut être déclarée en utilisant l'instruction *meta int remaining ;* dans le système d'automates. Des instructions d'affectation de la variable *remaining* peuvent être associées aux transitions dans l'automate. Cette variable sert à exprimer une heuristique estimant le coût restant après le déclenchement d'une transition.

Uppaal-Cora traite les requêtes d'atteignabilité exprimées dans la même syntaxe que dans Uppaal : $\exists\Diamond\varphi$ où φ désigne l'*configuration de l'automate* à atteindre. Cette requête désigne à parcourir toutes les exécutions menant à une configuration satisfaisant la propiété φ afin de trouver celle de coût minimum. Uppaal-Cora implémente un algorithme du type A^* [HNR68] pour effectuer la recherche du meilleur coût.

6.4 Synthèse de stratégie datée de fertilisation

Nous présentons la méthode développée pour effectuer une synthèse de stratégie datée de fertilisation pour une exploitation donnée. Cette méthode utilise l'automate temporisé de coût et le logiciel Uppaal-Cora présenté dans la section 6.3.

[1] CORA : Cost Optimal Reachability Anslysis

6.4.1 Stratégie datée de fertilisation

Une stratégie **datée** de fertilisation est un ensemble de *décisions de fertilisation* du type "appliquer telle quantité d'engrais sur telle parcelle à telle date". Les règles de fertilisation (cf la section 5.3.2.3) sont décrites comme suit :

1. Une dose d'engrais est fixée à une quantité définie par *DoseEngrais* ;
2. La quantité d'une application d'engrais est un multiple de cette dose.

Dans un premier temps, nous nous restreignons à une seule dose d'engrais par application pour simplifier le problème. Le problème consiste donc à décider quelle parcelle à fertiliser et à quelle date. Étant donné une fonction de coût, UPPAAL-CORA doit trouver une des meilleurs exécutions de pâturage minimisant cette fonction. Les décisions de fertilisation (fertiliser ou ne pas fertiliser une parcelle) tout au long de cette exécution forment une stratégie optimale de fertilisation.

6.4.2 Fonction de coût

Nous avons décidé, dans un premier temps, d'utiliser une fonction de coût qui prend en compte la consommation de fourrage conservé et l'utilisation d'engrais qui posent respectivement un problème économique et un problème environnemental. Concrètement la fonction utilisée est :

$$coût = prixFourrage * quantitéFourrage + prixEngrais * quantitéEngrais$$

Nous avons choisi la valeur 1 pour le paramètre *prixFourrage* et 10 pour *prixEngrais* en considérant non seulement que l'engrais est plus cher que fourrage conservé, et aussi que le problème apporté par l'application d'engrais est plus important. Les valeurs des paramètres ont été choisies de manière arbitraire pour la démonstration de la méthodologie. La fonction de coût peut facilement être adaptée pour tenir compte d'un contexte réel.

6.4.3 Modèle en automates temporisés de coût

Le modèle de pâturage a été modifié de la façon suivante pour permettre à UPPAAL-CORA de chercher le meilleur scénario de pâturage :

- Automate de contrôle de fertilisation (figure 6.3) : Une fois activé, l'automate bascule de l'état *Inactif* à l'état *Décider* et appelle la fonction *vérifierCondition* pour vérifier si les conditions de fertilisations sont satisfaites. Si oui, l'automate peut revenir à l'état *Inactif* en passant par la transition étiquetée par la garde "*condition==VRAI*". Passer par cette transition déclenche, par la synchronisation *Fertiliser !*, une action de fertilisation sur la parcelle contrôlée par cet automate et met à jour la variable de coût "*cost*" en ajoutant le coût de l'engrais appliqué. Dans l'étiquette *cost+=prixDoseEngrais*surface[parcelle]*, le paramètre *prixDoseEngrais* est égal à *prixEngrais * DoseEngrais* (paramètre renseigné dans le tableau 5.8). Une autre transition sans garde permet à l'automate de revenir à l'état *Inactif* quelle que soit la valeur de la variable *condition*. Cette caractéristique de

non déterminisme permet à UPPAAL-CORA d'explorer toutes les possibilités de décision.

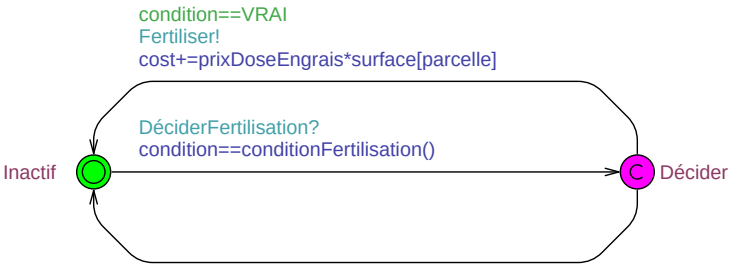


FIG. 6.3 – Automate de coût de contrôle de fertilisation

– Horloge centrale (figure 6.4) : Lorsque l'horloge centrale appelle la fonction *MettreÀJour* pour calculer le changement de la hauteur d'herbe pendant le passage de l'état *MiseÀJourPrairies*, elle récupère le montant de fourrage consommé par tous les troupeaux "*fourrage=MettreÀJour()*". Ce montant est ensuite ajouté à la variable de coût "*cost*" par l'instruction "*cost+=fourrage*".

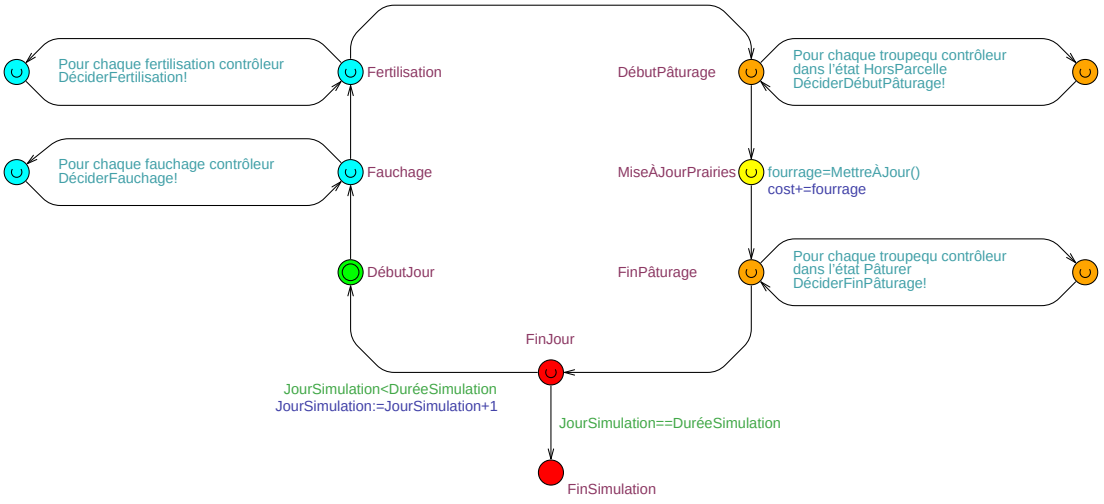


FIG. 6.4 – Automate de coût de l'horloge centrale

Dans notre travail, la variable de l'heuristique *remaining* n'est pas utilisée parce que nous n'avons pas d'information sur l'estimation de la consommation du fourrage conservé et de la fertilisation. Il n'est pas facile de définir facilement une heuristique.

6.4.4 Synthèse de stratégie par "Model-checking"

Une fois qu'un modèle de pâturage est construit à partir d'une description d'exploitation (voir la section 5.3) avec les modèles d'automates de coût, UPPAAL-CORA est utilisé pour trouver un des scénarios minimisant la fonction de coût. Il utilise deux

paramètres dont un est le modèle de l'exploitation en automates temporisés de coût et l'autre est la même requête en TCTL que celle décrite dans la section 5.4. UPPAAL-CORA retourne le coût optimal et une trace d'exécution qui correspond au coût minimal. Les *décisions de fertilisation* peuvent ensuite être extraites de la trace et forment une stratégie de fertilisation.

Cette méthode de synthèse de stratégie donne une solution précise et exacte (date, parcelle) pour une exploitation donnée. Si l'exploitation change, il faut générer un nouvel ensemble d'automates temporisés de coût et lancer de nouveau la recherche de stratégie optimale. Il y a donc manque de généricité. De plus cette solution strictement datée ne correspond pas aux habitudes des exploitants agricoles qui attendent une aide à leur gestion, mais pas un protocole précis indiquant exactement à quelle date et quelles parcelles ils doivent fertiliser. Cette approche est dans tous les cas inadaptée à une modélisation qui ne tient pas compte des conditions météorologiques, des autres contraintes d'exploitation par exemple jour de semaine/weekend etc. Enfin, à cause de l'explosion combinatoire, cette méthode est coûteuse en mémoire et ne peut synthétiser une stratégie que pour une durée relativement courte par rapport à la longueur d'une saison de pâturage.

6.4.5 Expérimentations

Sur une exploitation d'un troupeau de 50 animaux et une prairie de 12,5 ha divisée en sept parcelles numérotées de 1 à 7 dont la surface est comprise entre 1 ha et 4 ha, étant donné le 16 mars comme date de début de simulation et le 30 juin comme date de fin d'exécution, soit une durée de 107 jours, nous cherchons une stratégie optimale de fertilisation qui minimise la fonction de coût décrite dans la section 6.4.2. La sortie d'UPPAAL-CORA donne les dates et les parcelles à fertiliser :

- le 16 mars une dose sur la parcelle 3, 4, 6 et 7
- le 19 mars une dose sur la parcelle 1
- le 21 mars une dose sur la parcelle 2
- le 4 avril une dose sur la parcelle 5
- le 5 mai une dose sur la parcelle 1
- ne pas fertiliser même si l'application d'engrais est possible pour les autres dates

En appliquant cette stratégie, le coût s'élève à 23350 dont 18350 vient de la consommation de fourrage conservé (18350 kg) et 5000 vient de l'application d'engrais (500 kg N). Pour comparaison, une exécution où la fertilisation est interdite conduit à un coût de 47950 totalement dû à la consommation de fourrage conservé (47950 kg). Une autre exécution où la fertilisation est systématique après le départ de troupeau conduit à un coût de 42350 dont 24750 vient de la consommation de fourrage conservé (24750 kg) et 17600 vient de l'application d'engrais (1760 kg N).

6.5 Synthèse de stratégie générique de fertilisation

La méthode présentée dans la section 6.4 permet de générer une stratégie de fertilisation datée pour une exploitation donnée. Comme les exploitations peuvent être typées

en fonction de la taille des prairies, du nombre de parcelles et du nombre d'animaux etc, il est plus intéressant de chercher une stratégie générique qui s'applique à un type d'exploitation. Cela correspond mieux à ce qui est recherché par les experts. Les deux approches présentées dans cette section ont cet objectif et s'appliquent à recherche de stratégies de plus en plus génériques. Elles utilisent l'apprentissage supervisé dans ce but.

6.5.1 Stratégie complète d'une exploitation

Comme on l'a vu dans la section 6.4, la sortie d'UPPAAL-CORA fournit une et une seule trace d'exécution qui a un coût optimal. Les décisions de fertilisation extraites de cette trace forment une *stratégie minimale* (définition dans la section 4.4). Cependant d'autres exécutions ayant le coût optimal peuvent exister. UPPAAL-TIGA est utilisé pour extraire la *stratégie complète* (définition dans la section 4.4) à partir du coût optimal fourni par UPPAAL-CORA. Deux paramètres sont passés à UPPAAL-TIGA :

- les automates temporisés de coût générés à partir de la description de l'exploitation sauf que la variable *cost* doit être explicitement déclarée ;
- une requête de synthèse de contrôleur

$$control: \ A\,[not\ (cost > coûtOptimal)\ U\ Horloge.FinSimulation]$$

qui signifie "trouver la stratégie pour arriver à la fin de simulation sans que la valeur de la variable *cost* dépasse *coûtOptimal*".

On obtient une *stratégie complète* de fertilisation sous forme d'un ensemble de *décision de fertilisation*.

Exemple

Le tableau 6.1 montre la *stratégie complète* pour l'exploitation utilisée comme exemple en 6.4.5. Nous pouvons constater qu'il existe des décisions contradictoires pour une même situation. Ceci signifie que les deux choix pour cette situation sont tous possibles pour atteindre le coût optimal.

6.5.2 Apprentissage d'une stratégie générique

Pour construire une stratégie générique correspondant à un type d'exploitation, une méthode est d'étudier un ensemble de stratégies complètes pour différentes exploitations de même type et d'extraire les points communs entre ces stratégies complètes par un algorithme d'apprentissage supervisé. Des attributs supplémentaires décrivant le contexte de décision ont été ajoutés pour mieux décrire les décisions de fertilisation. La date de décision est remplacée par la décade. Le numéro de parcelle est remplacé par les attributs décrivant la parcelle suivants : le taux de croissance de l'herbe sur la parcelle, la surface de la parcelle, la hauteur d'herbe, l'hydromorphie du sol de la parcelle et la distance entre la parcelle et le bâtiment de traite. Finalement, un attribut le stock global d'herbe décrit le stock cumulé d'herbe sur toutes les parcelles de l'exploitation. Le tableau 6.2 montre des exemples de la base d'apprentissage.

Date	Parcelle	Décision de fertilisation
16 Mars	2	Oui
16 Mars	2	Non
16 Mars	3	Oui
16 Mars	3	Non
16 Mars	4	Oui
16 Mars	4	Non
16 Mars	5	Oui
16 Mars	6	Oui
19 Mars	1	Oui
21 Mars	2	Oui
23 Mars	3	Oui
25 Mars	4	Oui
4 Avril	5	Oui
5 Mai	1	Oui
5 Mai	1	Oui
Tous les autres cas		Non

TAB. 6.1 – Stratégie complète de fertilisation

Décade	Taux de croissance	Surface	Hauteur d'herbe	Hydro-morphie	Distance	Stock d'herbe	Décision
17	118	1,6	7,3	Non	Moyenne	18	Non
7	52	1,7	5,6	Non	Faible	5	Oui
9	99	1,8	4,9	Non	Moyenne	1	Oui

TAB. 6.2 – Extraits de la base d'apprentissage d'une stratégie générique

Les démarches de la synthèse de stratégie générique pour un type d'exploitation de pâturage sont décrites comme suivantes :

1. Générer un ensemble d'exploitations à partir du type d'exploitation donné ;
2. Pour chaque exploitation, produire la stratégie complète de fertilisation par la méthode présentée dans la section 6.5.1, enlever les paires de décisions contradictoires, remplir la base d'apprentissage avec les décisions de fertilisation accompagnées de son contexte de la décision ;
3. En spécifiant la décision de fertilisation comme l'étiquette de classe, appliquer un algorithme de classification pour extraire des règles qui forme une stratégie générique.

L'algorithme *JRip* a été retenu après avoir comparé plusieurs algorithme de classification. C'est un algorithme de classification de la famille *apprentissage de règles*. Dans

notre cas, l'interprétation d'une règle est de tester les attributs d'une condition de parcelle pour décider si il faut fertiliser. Ces règles sont faciles à appréhender et à utiliser par les agriculteurs.

6.5.3 Expérimentations

Nous avons essayé la méthode de synthèse présentée en 6.5.1 sur différentes exploitations de différentes caractéristiques (le nombre d'animaux, le nombre de parcelles, la durée de simulation etc). À cause de la capacité de calcul de logiciel UPPAAL, nous avons choisi « Petite 1 » (voir le tableau 5.1) comme type d'exploitation et 108 jours comme durée de simulation. Au delà de ce type d'exploitation (le nombre de parcelles précisément) et de la durée, le plantage à cause de débordement de mémoire est fréquent.

Expérimentation 1

La première expérimentation consiste à générer 100 exploitations à partir du type « Petite 1 » (voir la section 5.5). Ces exploitations ont été divisées en deux groupes avec 50 exploitations pour chaque groupe. Les décisions de fertilisation pour les 50 exploitations d'un groupe forment une base d'apprentissage. Celles de l'autre groupe servent de base de test.

Les règles apprises à partir de 1676 décisions des 50 exploitations du groupe de base d'apprentissage sont :

1. Fertiliser si décade ≤ 8 (201/82)
2. Fertiliser si décade ≤ 9 et la surface de la parcelle $\geq$ 1,8 ha (216/61)
3. Fertiliser si décade $\in [8, 9]$ et le stock d'herbe ≥ 4 jours (33/5)
4. Ne pas fertiliser dans tous les autres cas (1226/54)

Un test de règles a été fait avec 1735 décisions des 50 exploitations du groupe de test. Le résultat est montré dans le tableau 6.3. Les valeurs dans la colonne *Précision* montrent le pourcentage d'éléments correctement classifiés parmi tous les éléments classifiés dans cette classe. Les valeurs dans la ligne *Rappel* montrent le pourcentage d'éléments correctement classifiés parmi tous les éléments appartenant dans cette classe [OD08]. La valeur de *Kappa de Cohen* montre l'accord entre la classification par le classificateur et la vraie classe des éléments [Coh60]. Dans cette expérimentation, la valeur 0,699 montre un taux acceptable de cohérence.

	Ne pas fertiliser	Fertiliser	Précision
Prédiction ne pas fertiliser	1243	47	96,36%
Prédiction fertiliser	140	305	68,54%
Rappel	89,88%	86,65%	
Kappa de Cohen	0,697		

TAB. 6.3 – L'analyse de la performance des règles génériques de fertilisation

Pour évaluer la performance des règles obtenues, nous avons relancé les exécutions sur les 100 exploitations générées lors de la phase d'apprentissage, en utilisant les règles obtenues comme stratégie de fertilisation. Nous obtenons ainsi un coût réel lié au pâturage pour chaque exploitation. Nous avons mémorisé lors de la phase d'apprentissage le coût minimal pour chaque exploitation. Nous avons ensuite comparé ces coûts réels avec les coût minimals pour chaque exploitation. Le résultat de comparaison est présenté dans la figure 6.5. Dans 29% des cas, le coût réel dépasse de moins de 10% le coût minimal. Dans 38% des cas, le dépassement est entre 10% et 20% le coût minimal. Dans 25% des cas, le dépassement est compris entre 20% et 30% le coût minimal. Dans 7% de cas, le dépassement s'élève entre 30% et 40% le coût minimal. Seulement dans 1% des cas, le coût réel dépasse de plus de 40% le coût minimal. En moyen le dépassement du coût minimal est 16%, ce qui montre que les règles sont suffisamment bonnes pour conduire le pâturage vers le coût minimal.

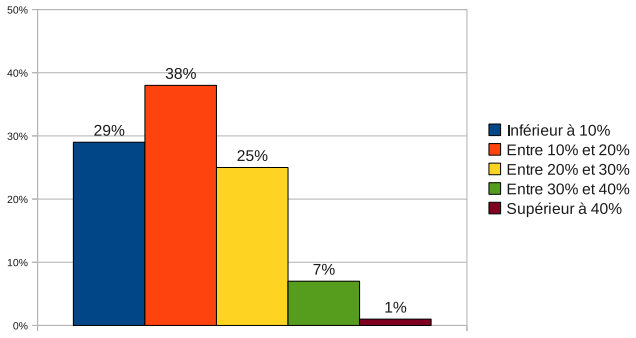


FIG. 6.5 – Résultat des vérifications des règles générées sur 100 exploitations.

Expérimentation 2

Dans la deuxième expérimentation, nous avons utilisé la même procédure que dans la première expérimentation sauf que le climat pendant le mois de mars, d'avril et de mai est *Défavorable*. Les règles apprises à partir de 1753 décisions des 50 exploitations du groupe de base d'apprentissage sont :

1. Fertiliser si décade ≤ 8 et la distance de la parcelle est moyenne et la surface de parcelle $\geq$ 2,5 ha (14/0)
2. Fertiliser si décade ≤ 8 et la distance de la parcelle est moyenne et la surface de parcelle = 1,9 ha (9/0)
3. Fertiliser si décade ≤ 8 et la distance de la parcelle est moyenne et la surface de parcelle = 1,7 ha et stock de l'herbe $\geq$ 14 (14/1)
4. Fertiliser si décade = 8 et surface de parcelle = 16 (45/0)
5. Fertiliser si décade ≤ 7 et la distance de la parcelle est moyenne et la surface de parcelle = 1,6 ha (10/1)

6. Fertiliser si décade ≤ 7 et la distance de la parcelle est moyenne et la surface de parcelle $\in$ [2,0 2,2] ha et stock de l'herbe ≤ 7 jours (14/2)
7. Fertiliser si décade ≤ 7 et la distance de la parcelle est moyenne et la surface de parcelle $\geq$ 1,5 ha et stock de l'herbe ≤ 7 jours (22/3)
8. Fertiliser si décade ≤ 9 et la surface de parcelle $\in$ [1,5 1,8] ha et la distance de la parcelle est faible et stock de l'herbe ≥ 11 jours (14/1)
9. Fertiliser si décade ≤ 9 et la distance de la parcelle est moyenne et stock de l'herbe ≥ 14 jours et la surface de parcelle = 1,6 ha (10/1)
10. Fertiliser si décade ≤ 7 et la distance de la parcelle est moyenne et la surface de parcelle $\geq$ 1,8 ha (38/9)
11. Fertiliser si décade ≤ 9 et la hauteur de l'herbe $\leq$ 5,6 cm et stock de l'herbe ≤ 2 jours et la surface de parcelle $\leq$ 2,4 ha (9/0)
12. Fertiliser si décade ≤ 9 et la distance de la parcelle est moyenne et la surface de parcelle $\geq$ 1,5 ha et stock de l'herbe ≥ 11 jours (11/2)
13. Fertiliser si décade = 8 et stock de l'herbe $\in$ [4 6] jours (20/2)
14. Fertiliser si décade ≤ 9 et la surface de parcelle $\in$ [2,0 2,2] ha et stock de l'herbe ≥ 11 jours (15/4)
15. Fertiliser si décade ≤ 9 et la distance de la parcelle est moyenne et la surface de parcelle $\geq$ 1,1 ha et stock de l'herbe ≥ 11 jours (14/4)
16. Fertiliser si décade ≤ 9 et la hauteur de l'herbe $\in$ [5,3 5,5] ha et stock de l'herbe ≥ 4 jours (16/3)
17. Fertiliser si décade ≤ 9 et la distance de la parcelle est moyenne et stock de l'herbe ≥ 4 jours (28/11)
18. Fertiliser si décade ≤ 8 et la la surface de parcelle $\in$ [2,1 2,3] ha (17/5)
19. Ne pas fertiliser dans tous les autres cas (1433/85)

Un test de règles a été fait avec 1773 décisions des 50 exploitations du groupe de test. Le résultat est montré dans le tableau 6.4. La valeur 0,631 de *Kappa de Cohen* montre un taux acceptable de cohérence.

	Ne pas fertiliser	Fertiliser	Précision
Prédiction ne pas fertiliser	1346	132	91,07%
Prédiction fertiliser	66	229	77,63%
Rappel	95,33%	63,43%	
Kappa de Cohen	0,631		

TAB. 6.4 – L'analyse de la performance des règles génériques de fertilisation

Nous avons fait la même évaluation comme dans l'expérimentation 1. Le résultat de comparaison est présenté dans la figure 6.5. Dans 14% des cas, le coût réel dépasse de moins de 10% le coût minimal. Dans 23% des cas, le dépassement est entre 10% et 20%

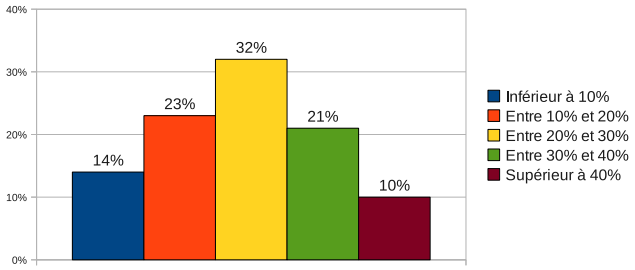


FIG. 6.6 – Résultat des vérifications des règles générées

le coût minimal. Dans 32% des cas, le dépassement est compris entre 20% et 30% le coût minimal. Dans 21% de cas, le dépassement s'élève entre 30% et 40% le coût minimal. Dans 10% des cas, le coût réel dépasse de plus de 40% le coût minimal. En moyen le dépassement du coût minimal est 23%, ce qui montre que les règles sont bonnes pour conduire le pâturage vers le coût minimal.

6.6 Synthèse de méta-stratégie

La stratégie générique présentée dans la section 6.5 convient mieux aux habitudes des exploitants agricoles en comparaison avec la stratégie datée. Ce type de stratégie permet de décider la fertilisation sur une parcelle par vérification de situation de la parcelle avec les règles décrites dans la stratégie. Par contre, quand le nombre de règles est important, la vérification à la main peut être difficile. De plus, ce type de stratégie n'offre toujours pas de flexibilité parce que la stratégie demande qu'une fertilisation doit être effectuée dès qu'une condition est remplie. Nous proposons un autre type de stratégie : méta-stratégies qui offrent plus de flexibilité et sont faciles à utiliser.

6.6.1 Définitions

Méta-données

Nous utilisons des méta-données pour décrire les scénarios de pâturage de manière statistique et abstraite. Par exemple "20% de surface des prairies a été fertilisé pendant la première décade". Dans un premier temps, nous avons utilisé un quintuplet de méta-données qui sont :

- le pourcentage de surface fertilisée pendant chaque décade
- la charge de prairies fertilisées pour le pâturage en nbAnimaux*heure
- la charge de prairies non fertilisées pour le pâturage en nbAnimaux*heure
- la surface en ha de prairie fauchée et fertilisée
- la surface en ha de prairie fauchée mais non fertilisées

Méta-stratégie

Une méta-stratégie propose une actions à partie des propriétés sur les méta-données alors que la stratégie classique propose une action en fonction de la condition globale au moment de faire la décision. Par exemple, une propriété proposée par la méta-stratégie peut être "fertiliser au plus 20% de surface de prairie pendant la première décade de pâturage et assurer que la charge de prairie fertilisée pour le pâturage soit supérieure que 1000 animaux*heure".

6.6.2 Synthèse de méta-stratégie

Nous utilisons l'apprentissage automatique pour générer une méta-stratégie pour un type d'exploitation donné. Pour cela, nous avons besoin d'une base d'apprentissage composée des quintuplets de méta-données. Chaque quintuplet décrit un scénario de pâturage et est étiqueté par le coût de pâturage.

Étant donné un type de d'exploitation, un ensemble d'exploitations est généré à partir de ce type donné. Pour chaque exploitation :

1. À l'aide de l'UPPAAL-CORA, trouver une des exécutions optimale de pâturage.
2. Extraire le quintuplet de méta-données à partir de l'exécution optimale trouvée.
3. Énumérer toutes les exécutions dont le pourcentage de surface fertilisée pendant chaque décade correspond approximativement ($\pm 1\%$) à celui de la meilleure exécution. Extraire les quintuplets de méta-données de toutes les exécutions énumérées, associés au coût de pâturage.

Les coûts de pâturage de différentes exploitations ne sont pas comparables. Par exemple, le coût optimal de pâturage d'une exploitation peut être double de celui d'une autre exploitation. Pour pouvoir les comparer, le coût de pâturage est transformé en une valeur catégorique de la façon suivante :

1. Pendant l'étape d'énumération d'exécutions présentée dessus 3, mémoriser les coûts de pâturage de toutes les exécutions ;
2. Trouver l'intervalle $[c_{min},\ c_{max}]$ du coût de pâturage ;
3. Pour chaque quintuplet, si le coût de pâturage associé se trouve dans l'intervalle $[c_{min},\ c_{min} + 0.2 * (c_{max} - c_{min})]$, étiqueter le quintuplet *Bon*. Si le coût se trouve dans l'intervalle $[c_{max} - 0.2 * (c_{max} - c_{min}),\ c_{max}]$, étiqueter le quintuplet *Mauvais*. Si le coût ne se trouve pas dans les deux intervalles, le quintuplet est enlevé de la base.

Cette étiquette permet à un algorithme de classification de trouver des règles pour classifier les quintuplets. L'algorithme de classification *JRip* a été utilisé pour générer les règles pour séparer les *Bons* et les *Mauvais*.

6.6.3 Expérimentations

50 exploitations ont été aléatoirement générées à partir du type d'exploitation « Petite 1 » décrit dans le tableau 5.1. Étant donné la date du le 16 avril comme début

de simulation et 108 comme durée de simulation, les 1331 quintuplets de méta-données générés à partir des exploitations forment une base d'apprentissage. Voici un extrait :

- 17,1% de surface de prairie a été fertilisée pendant la 11ème décade. La charge de prairie fertilisée s'est élevée à 22000. La charge de prairie non fertilisée s'est élevée à 86000. Aucun fauchage n'a été effectué. Le coût de pâturage a été évalué *Bon.*
- 14,2% de surface de prairie a été fertilisée pendant la 11ème décade. 4,6% de surface de prairie a été fertilisée pendant la 12ème décade. La charge de prairie fertilisée s'est élevée à 19000. La charge de prairie non fertilisée s'est élevée à 89000. Aucun fauchage n'a été effectué. Le coût de pâturage a été évalué *Bon.*
- 13,6% de surface de prairie a été fertilisée pendant la 11ème décade. 4,6% de surface de prairie a été fertilisée pendant la 12ème décade. La charge de prairie fertilisée s'est élevée à 14000. La charge de prairie non fertilisée s'est élevée à 94000. 8,6% de surface de prairie non fertilisée a été fauché. Le coût de pâturage a été évalué *Mauvais.*

À la sortie de l'algorithme *JRip*, nous avons récupéré les règles formant une méta-stratégie :

- Si la charge de prairies fertilisées pour le pâturage ≥ 17000 ;
- ou la charge de prairies fertilisées pour le pâturage ≥ 14000 et la charge de prairies fertilisées pour le pâturage ≥ 92000 et la surface de prairie fertilisée fauchée $\leq 13{,}9\%$;

le coût de pâturage est proche du coût optimale et classé *Bon.* Sinon le coût de pâturage est élevé et classé *Mauvais.*

50 autres exploitations ont été générées pour créer une base de test de 1831 quintuplets. Le résultat d'analyse de performance est montré dans le tableau 6.5. La valeur de *Kappa de Cohen* 0,722 montre une forte cohérence.

	Bonne	Mauvaise	Précision
Prédiction bonne	373	64	85,35%
Prédiction mauvaise	30	235	88,68%
Rappel	92,56%	79,60%	
Kappa de Cohen	0,722		

TAB. 6.5 – L'analyse de la performance de méta-stratégie

Pour utiliser une méta-stratégie, il suffit de choisir une des conditions listées et de planifier les activités de pâturage selon cette condition. Par exemple si on prend la première condition de la stratégie : la charge de prairies fertilisées pour le pâturage ≥ 17000, il faut planifier la fertilisation et la mise au pâturage pour que le troupeau pâturage pendant au moins 340 heures sur les parcelles fertilisées (17000 = 340 heures * 50 animaux). Cela laisse une grande flexibilité aux exploitants pour générer leurs ressources (prairies, main-d'œuvres et machines etc).

6.7 Conclusion

Nous avons présenté les méthodes de synthèse de stratégies des actions de pâturages dans ce chapitre. Les méthodes développées sont les suivantes :

La première méthode est la synthèse de stratégie optimale par la technique de "générer et tester" pour la recherche de stratégie de la mise au pâturage. La stratégie de la mise au pâturage est paramétrable. La technique de "générer et tester" a été choisie parce que les paramètres de stratégie ont des valeurs discrètes. Cette méthode s'appuie sur le logiciel PaturMata grâce à sa performance d'exécution.

La deuxième méthode est la synthèse de stratégie optimale par la technique de *model-checking* pour la synthèse de stratégie datée de la fertilisation. Le formalisme de l'automate temporisé de coût a été utilisé pour exprimer le coût de pâturage. Ce formalisme permet à l'algorithme de *model-checking* d'effectuer non seulement la vérification d'atteignabilité mais aussi la recherche de meilleur chemin d'exécution. Uppaal-Cora implémente ce formalisme et a été utilisé dans la méthode. Étant donné une exploitation de pâturage, cette méthode donne une stratégie datée du type "à telle date fertiliser telle parcelle". Ce type de stratégie datée de fertilisation s'est avéré inadapté pour une utilisation dans la pratique. Les exploitants agricoles préfèrent une aide à leur gestion, mais pas un protocole précis indiquant exactement à quelle date et quelles parcelles ils doivent fertiliser. Mais cette technique est importante car elle est la base des deux méthodes suivantes, qui combinent toutes les deux synthèse de contrôleur et apprentissage supervisé.

La troisième méthode combine technique de *model-checking* et apprentissage automatique pour identifier des stratégies génériques de fertilisation. Cette méthode s'intéresse à synthétiser des règles génériques de fertilisation pour un type d'exploitation. Après avoir construit une base d'apprentissage avec les stratégies optimales d'un ensemble d'exploitation d'un type donné, l'algorithme d'apprentissage de règles *JRip* est utilisé pour extraire des règles de fertilisation. La stratégie obtenue est du type "si telles conditions sont remplies, fertiliser une parcelle". Les conditions portent sur la décade, la surface de parcelle, la hauteur d'herbe et le type de prairie etc. Ce type de stratégie est plus adapté aux exploitations agricoles car elle est plus facile à appréhender. Cependant, les règles de fertilisations dans une stratégie générique sont peu flexibles car elles demandent à fertiliser une parcelle dès que les conditions sont remplies. Nous avons développé un autre type de stratégie pour répondre mieux à la flexibilité.

La dernière méthode est la synthèse de méta-stratégie par la technique de *model-checking* et l'apprentissage automatique. Une méta-stratégie décrit les caractéristiques que les scénarios de pâturage doivent satisfaire. Par exemple, "fertiliser au moins 20% de la surface de prairie pendant le mois de mai". En conséquence, la méta-stratégie offre la meilleure flexibilité car les exploitants peuvent planifier librement leurs activités. Cette méthode utilise la technique de *model-checking* pour construire une base d'apprentissage avec les caractéristiques des scénarios de pâturage étiquetés par le coût de pâturage. *JRip* est ensuite utilisé pour produire une méta-stratégie qui est composée des caractéristiques communes des scénarios de pâturage de faible coût.

Les trois dernières méthodes de synthèse de stratégie présentées dessus s'appuient

sur la technique de la *synthèse de contrôleur* basée sur *model-checking*. Cette technique nécessite une quantité énorme de mémoire à cause de l'explosion combinatoire. Ceci nous oblige de travailler sur les petites exploitations de pâturage et pour une courte durée de simulation par rapport à une saison de pâturage. Cependant l'utilisation des techniques d'apprentissage supervisé combinée à la synthèse de contrôleur ouvre des perspective intéressantes développées dans les deux dernières méthodes.

Quatrième partie

Conclusion

Chapitre 7

Conclusion

Problématique

Dans le cadre de cette thèse, nous nous intéressons à la question de l'aide à la décision par la conception de modèles qualitatifs qui répondent de manière plus synthétique à des problèmes complexes dans le domaine de la gestion des agro-écosystèmes. Cette représentation se situe à un niveau d'abstraction plus élevée que ce qui est fait dans les modèles numériques, et est bien adaptée au cas où des utilisateurs veulent comparer plusieurs scénarios, et prendre des décisions à partir des informations issues des exécutions du modèle.

Le formalisme que nous proposons d'utiliser est celui des systèmes à événements discrets. Il est bien adapté au cas où les différentes entités interagissent entre elles, et où la dynamique dépend fortement des interactions mutuelles entre les entités. Par exemple, dans le domaine de la gestion de pêche, les entités sont les espèces de poissons, les pressions de pêche et les impacts environnementaux. Ceci permet de profiter de l'efficacité du model-checking. L'intérêt de l'utilisation du model-checking est de pouvoir vérifier des propriétés sous forme d'atteignabilité et/ou de sûreté du modèle au lieu d'exécuter de façon itérative le modèle.

Dans ce travail, nous nous sommes en particulier focalisés sur la recherche de stratégies optimales pour la gestion d'un agro-écosystème modélisé en automates temporisés. Nous avons exploré les différentes manières d'utiliser la technique de synthèse de contrôleur proposée par l'approche du model-checking. Dans le cadre de l'aide à la décision, les stratégies générées doivent être faciles à appréhender et à utiliser par les gestionnaires.

Apports de la thèse

Nous avons étudié deux problèmes dans deux domaines différents. Le premier problème, dans le domaine de l'halieutique, concerne la génération de stratégies de pêche. Le modèle EcoMata propose une modélisation qualitative d'un réseau trophique de plusieurs espèces de poissons sous pressions environnementales (changement de température, ouragan) et anthropiques (activités de pêche). Ce modèle est exprimé comme un

système à événements discrets sous forme d'un ensemble d'automates temporisés. Des scénarios de requêtes de type prédictif exprimés dans un langage de haut niveau sont également proposés et permettent aux utilisateurs d'interroger le modèle. Les scénarios sont interprétés comme des propriétés de sûreté et/ou d'atteignabilité exprimées en formules TCTL. Le model-checking est utilisé pour vérifier les propriétés sur le modèle d'automates temporisés.

Mes contributions principales au projet EcoMata sont : (1) l'amélioration de la génération automatique du modèle en automates temporisés à partir d'une description d'un réseau trophique sous forme d'un ensemble de paramètres des équations de Lotka-Volterra ; (2) le développement de deux méthodes de génération de stratégies optimales de pêche. La première méthode utilise la synthèse de contrôleur sur les automates temporisés de jeux. La stratégie générée est de type "dans telle condition, appliquer telle action". Cependant, dans le domaine halieutique, il est difficilement envisageable de surveiller en continu l'écosystème, ce qu'exigerait pourtant la "condition", des stratégies obtenues par cette méthode. Une deuxième méthode de type "générer et tester" a été développée. Cette méthode est basée sur l'énumération de plannings de pêche qui ne dépend que du calendrier et qui correspond mieux aux pratiques des gestionnaires de pêche ; (3) le développement d'une interface graphique sur laquelle les utilisateurs peuvent saisir la description de réseau trophique, lancer les requêtes pour explorer le réseau, visualiser les résultats et comparer les stratégies.

Le second problème, dans le domaine de l'agronomie, concerne l'aide à la gestion du pâturage au sein d'une exploitation agricole. Nous avons tout d'abord proposé une modélisation des activités liées au pâturage, et, dans un second temps, des méthodes de recherche de stratégies optimales de ces activités. La modélisation proposée représente également les activités liées au pâturage dans le formalisme des automates temporisés. Le modèle peut représenter la dynamique de l'herbe sur les parcelles en prairie, les activités liées au pâturage et la prise de décision des activités selon des stratégies. Nous avons conçu une modélisation hybride, nommée PaturMata, qui implémente cette approche. Les sous-modèles de PaturMata sont organisés dans une hiérarchie composée en quatre couches pour mieux séparer les rôles dans le modèle : (1) la *couche prairie* représente la dynamique de l'herbe dans les parcelles en prairie. Les variables numériques sont utilisées afin de représenter la dynamique de l'herbe avec une meilleure précision, ce qui donne la particularité du caractère hybride de ce modèle ; (2) les automates temporisés dans la *couche exécution* représentent les activités liées aux pâturage ; (3) les automates dans la *couche contrôleur* représentent le processus de prise de décision des activités de pâturage selon une stratégie ; (4) l'*horloge centrale* représente l'avancement du temps.

Nous avons ensuite étudié la recherche des stratégies optimales des activités liées au pâturage : la mise au pâturage, le fauchage et la fertilisation. Comme la stratégie de fauchage suit des règles simples, nous avons particulièrement traité la recherche de stratégies optimales de la mise au pâturage et de la fertilisation. En tenant compte des pratiques réelles, nous sommes partis d'une stratégie de mise au pâturage très utilisée, et

avons développé une méthode de recherche de stratégies optimales basée sur la technique "générer et tester". Les utilisateurs peuvent choisir la meilleure stratégie en comparant les résultats des exécutions.

Pour la recherche des stratégies optimales de fertilisation, nous avons développé trois méthodes.

- La première méthode transforme le modèle de pâturage en automates temporisés de coût pour représenter le coût lié au pâturage tout au long de l'exécution. L'algorithme de synthèse de contrôleur est utilisé pour synthétiser une stratégie optimale datée de fertilisation qui minimise le coût lié au pâturage. La stratégie générée est de type "à telle date, faire telle action". La stratégie est spécifique à une exploitation donnée et n'a pas de généricité. Lorsque l'exploitation change, il est nécessaire de recommencer la recherche de stratégies optimales.
- La deuxième méthode consiste à générer des stratégies optimales pour un type d'exploitation. La méthode cherche d'abord les stratégies optimales d'un ensemble d'exploitations de même type. Un algorithme d'apprentissage automatique est ensuite utilisé pour extraire des stratégies génériques de fertilisation, sous forme d'un ensemble de règles, pour ce type d'exploitation. Les règles sont de type "si telles conditions sont vérifiées, faire telle action". Ce type de stratégie demande l'application d'une action dès que les conditions sont vérifiées, mais ne prend pas en compte les conditions climatiques ni la date calendaire.
- La troisième méthode génère des méta-stratégies. La méta-stratégie est un type de stratégie abstraite qui exprime les règles de manière globale. Par exemple, "fertiliser 20% de la surface totale de la prairie pendant la deuxième décade d'avril". Ce type de stratégie permet aux exploitants d'organiser librement leurs activités, selon le contexte précis d'applications.

Ces différentes méthodes des recherche de stratégies s'appuient toutes sur un model-checker afin d'explorer le modèle décrit par automates temporisés. Dans notre travail, nous avons choisi le model-checker Uppaal, spécialisé pour la vérification d'automates temporisés. À la fin de l'exécution, des informations concernant les activités de pâturage peuvent être extraites à partir d'une trace fournie par Uppaal. Ces informations permettent de calculer des indicateurs afin d'évaluer les stratégies des activités liées au pâturage. Autour de ces techniques, un logiciel prototype nommé PaturMata a été développé. Ce logiciel propose une interface graphique conviviale permettant aux utilisateurs de saisir les informations décrivant une exploitation et de vérifier la validité des stratégies sur le modèle. Il récupère la trace d'exécution et présente le résultat sous forme graphique. PaturMata implémente également la méthode de la recherche de stratégies optimales de mise au pâturage.

Les deux approches proposées dans EcoMata et PaturMata sont relativement génériques et sont applicables à tout système composé d'un ensemble d'entités dont la dynamique dépend des interactions entre elles.

Dans le cas de l'halieutique, les entités sont les espèces de poissons, les impacts environnementaux et les pressions de pêche. La dynamique des biomasses de chaque espèce de poisson dépend des relations proie-prédateur avec les autres espèces, des impacts environnementaux et des pressions anthropiques. En fait tout réseau trophique de type proie-prédateur peut être modélisé dans EcoMata. Les données d'un réseau halieutique peuvent être imprécises voire inconnues, ce qui rend difficile toute modélisation numérique reposant sur de nombreux paramètres biologiques. La modélisation qualitative d'EcoMata offre la possibilité de représenter ces systèmes mal connus à l'aide d'une incertitude temporelle qui rend les automates non déterministes. Cette caractéristique répond mieux à la question de modélisation mais augmente la complexité de l'exécution du modèle.

Dans l'autre cas, celui de gestion de pâturage, les entités sont les parcelles de prairie, les troupeaux et les exploitants. La dynamique de l'herbe sur les parcelles dépend de la présence des animaux (broutage) et des actions des exploitants (fauchage, fertilisation). Or, l'herbe sur les parcelles a une propre dynamique à l'échelle journalière qui demande qu'elle soit représentée avec une précision suffisante. Ceci nécessite un passage d'une modélisation en automates temporisés à une modélisation hybride, qui combine un modèle numérique et un modèle qualitatif. Ce passage nous a aussi conduit à concevoir un méta-modèle avec quatre couches dans lequel on met bien en évidence les entités, les contrôles et l'horloge. Par ailleurs, PaturMata est déterministe, mais rien n'empêcherait de représenter le fonctionnement d'une exploitation non déterministe.

Le tableau 7.1 présente une comparaison des deux modélisations.

	Déterministe	Incertitude Temporelle	Hybride	Hiérarchique	Granularité de temps
EcoMata	Non	Oui	Non	Non	Mois, année
PaturMata	Oui/Non	Non	Oui	Oui	Jour

Tab. 7.1 – Comparaison des modélisations

Perspectives

Les perspectives de ces travaux sont les suivantes :

Perspectives à court terme

En ce qui concerne EcoMata, nous envisageons de tester la méthode de recherche de stratégies optimales par la technique "générer et tester" sur un réseau trophique marin de taille réelle, par exemple celui d'Ouvéa [BGK04a]. L'automate correspondant possède 1652 localités et 20412 transitions. Le passage à l'échelle d'une application réelle permettra de tester les limites de la recherche de stratégies et éventuellement d'envisager des évolutions de mise en œuvre.

Nous envisageons également de donner le moyen aux utilisateurs de définir leurs propres méthodes de tri de plannings de pêche à l'aide d'une fonction de coût. Ceci permettra de prendre en compte les facteurs économiques en favorisant, par exemple, la pêche d'espèces aux périodes où le prix de vente est le plus intéressant.

En ce qui concerne PaturMata, la méthode de la recherche de stratégie générique de fertilisation présentée dans la section 6.5 nécessite la manipulation d'une chaîne d'outils : Uppaal-Cora, Uppaal-Tiga et RapidMiner. Une première et rapide évolution consisterait à automatiser l'enchaînement de ces outils. Dans un deuxième temps, il serait bénéfique d'automatiser la récupération des règles issues d'apprentissage et leurs transformations en automates temporisés afin de vérifier leur optimalité.

À cause des limitations du model-checker Uppaal-Cora, nous ne pouvons générer des stratégies optimales pour une durée limitée de 108 jours alors qu'une saison de pâturage dure environ 250 jours. Nous envisageons d'améliorer la structure d'automates temporisés ou d'ajouter une fonction heuristique pour que l'algorithme de recherche de stratégies optimales qui s'appuie sur A^* soit plus efficace.

PaturMata s'adresse à des utilisateurs de terrain et a été aujourd'hui seulement testé par des utilisateurs privilégiés. La diffusion du logiciel auprès d'une plus large communauté permettrait de faire ressortir la pertinence des résultats, et éventuellement l'adaptation à leur usage.

Perspectives moyen terme

Tout au long du travail de cette thèse, une des difficultés que nous ayons rencontrées est la performance des outils de synthèse de contrôleur. La complexité en espace mémoire de l'algorithme de synthèse de contrôleur augmente très rapidement en fonction de la taille du problème traité. À cause de la limite des outils existants, nous ne pouvons traiter que des modèles de taille limitée qui s'avère insuffisante pour la représentation d'application réelles. Nous espérons des nouveaux outils dans ce domaine ou des améliorations des outils existants afin d'augmenter la capacité de traitement de notre modélisation.

PaturMata repose sur une modélisation hybride de pâturage qui combine un modèle numérique et un modèle qualitatif en automates temporisés. Nous avons choisi Uppaal pour représenter les automates temporisés. Uppaal peut traiter des automates temporisés dans lesquels on utilise des variables numériques mais leur l'exploitation n'est pas efficace. Il serait intéressant d'utiliser des outils spécialisés pour la vérification de modèles hybrides. Ceci pourrait augmenter la performance de l'exécution du modèle.

La dernière version d'Uppaal propose une nouvelle fonctionnalité qui permet d'utiliser plusieurs horloges de différents pas. Cette fonctionnalité est bien adaptée à une modélisation temporelle multi-échelle. Dans le cadre de notre application à la gestion de pâturage, cette capacité permettrait de gérer la croissance de l'herbe de manière qualitative. Pour la modélisation des agro-écosystèmes en général, la représentation de différentes échelles de temps permettrait d'élargir les modèles susceptibles d'être représentés par notre méthode.

Une dernière piste consisterait à enployer une modélisation probabiliste telle que proposée par UPPAAL-PRO[1]. Dans cette approche, le model-checker détermine non seulement si un état est atteignable ou non mais également quelle est la probabilité d'atteindre cet état. Dans le cas de la représentation d'écosystème non déterministes pour lesquels nous disposons de probabilité sur les évolutions, cette faculté permettrait d'enrichir les résultats proposés aux utilisateurs.

[1] `http://people.cs.aau.dk/~arild/uppaal-probabilistic/`

Bibliographie

[ABC+05] K. Altisen, P. Bouyer, T. Cachat, F. Cassez, and G. Gardey. Introduction au contrôle des systèmes temps-réel. *European Journal of Automation*, 39(1-2-3) :367–380, 2005.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical computer science*, 126 :pp. 183–235, 1994.

[AGOS+09] P. Aurousseau, C. Gascuel-Odoux, H. Squividant, R. Trépos, F. Tortrat, and M.-O. Cordier. A plot drainage network as a conceptual tool for the spatial representation of surface flow pathways in agricultural catchments. *Computers and Geosciences*, 35 :276–288, 2009.

[AMBD98] M. Ajmone Marsan, A. Bobbio, and S. Donatelli. Petri nets in performance analysis : An introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I : Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 211–256. Springer Berlin Heidelberg, 1998.

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. 1998.

[BBF+01] B. Berard, M. Bidoit, A. Finkel, F. Laroussine, A. Petit, P. Schnoebelen, and P. McKenzie. *Systems and Software Verification. Model-Checking Techniques and Tools*. Springer-Verlag, 2001.

[BCQG11] T. Bouadi, M.-O. Cordier, R. Quiniou, and C. Gascuel. Analyse multidimensionnelle interactive de résultats de simulation d'un agro-système. In *INFORSID*, Lille, France, 2011.

[BDD+01] J.-E. Bergez, P. Debaeke, J.-M. Deumier, B. Lacroix, D. Leenhardt, P. Leroy, and D. Wallach. MODERATO : an object-oriented decision tool for designing maize irrigation schedules. *Ecological Modelling*, 137 :43–60, 2001.

[BDR+02] V. Beaujouan, P. Durand, L. Ruiz, P. Aurousseau, and G. Cotteret. A hydrological model dedicated to topography-based simulation of nitrogen transfer and transformation : rationale and application to the geomorphology - denitrification relationship. *Hydrological Processes*, 16(2) :493–507, 2002.

[BGK04a] Y.-M. Bozec, D. Gascuel, and M. Kulbicki. Trophic model of lagoonal communities in a large open atoll (uvea, loyalty islands, new caledonia). *Aquatic Living Resources*, 17 :151–162, 2004.

[BGK04b] Y.M. Bozec, D. Gascuel, and M. Kulbicki. Trophic model of lagoonal communities in a large open atoll (Uvea, Loyalty islands, New Caledonia). *Aquatic Living Resources*, 17 :151–162, 2004.

[BGL04] J. E. Bergez, F. Garcia, and L. Lapasse. A hierarchical partitioning method for optimizing irrigation strategies. *Agricultural Systems*, 80(3) :235–253, 2004.

[BLR05] G Behrmann, K.-G. Larsen, and J.-I. Rasmussen. Formal methods for components and objects. volume 3657 of *Lecture Notes in Computer Science*, pages 162–182. Springer Berlin Heidelberg, 2005.

[CB91] P. Clark and R. Boswell. Rule induction with cn2 : Some recent improvements. pages 151–163. Springer-Verlag, 1991.

[CDF+05] F. Cassez, A. David, E. Fleury, K.-G. Larsen, and D. Lime. Concur 2005 - concurrency theory. chapter Efficient on-the-fly algorithms for the analysis of timed games, pages 66–80. Springer-Verlag, London, UK, UK, 2005.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8 :244–263, 1986.

[CGG+05] M.-O. Cordier, F. Garcia, C. Gascuel, V. Masson, J. Salmon-Monviola, F. Tortrat, and R. Trépos. A machine learning approach for evaluating the impact of land use and management practices on streamwater pollution by pesticides. In Modelling, Simulation Society of Australia, and New Zealand, editors, *MODSIM'05 (International Congress on Modelling and Simulation)*, Melbourne, Australie, 2005.

[CGMC99] M. Cros, F. Garcia, and R. Martin-Clouaire. Sepatou : a decision support system for the management of rotational grazing in a dairy production. In *Proceedings of 2nd European Conference on Information Technology in Agriculture, in : Schiefer G., Helbig*, pages 549–557, 1999.

[CGMCD01] M. Cros, F. Garcia, R. Martin-Clouaire, and M. Duru. Simulation optimization of grazing management strategies, 2001.

[CGP02] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model-Checking*. MIT Press, 2002.

[CJL+09] F. Cassez, J.-J. Jessen, K.-G. Larsen, J.-F. Raskin, and P.-A. Reynier. Automatic synthesis of robust and optimal controllers — an industrial case study. pages 90–104, 2009.

[Coh60] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1) :37–46, 1960.

[DPF01] L. Delaby, J.L. Peyraud, and P. Faverdin. Patur' in : le paturage des vaches laitieres assiste par ordinateur. *Fourrages*, 167(2) :385–398, 2001.

[GD01] F. Guerrin and J. Dumas. Knowledge representation and qualitative simulation of salmon redd functioning. part i : qualitative modeling and simulation. *Biosystems*, 59(2) :75–84, 2001.

[GN72] R. Garfinkel and G.L. Nemhauser. *Integer programming*. Series in decision and control. Wiley, 1972.

[GOAC+09] C. Gascuel Odoux, P. Aurousseau, M.-O. Cordier, P. Durand, F. Garcia, V. Masson, J. Salmon-Monviola, F. Tortrat, and R. Trépos. A decision-oriented model to evaluate the effect of land use and agricultural management on herbicide contamination in stream water. *Environmental Modelling and Software*, 24 :1433–1446, 2009.

[Hél03] A. Hélias. *Agrégation/abstraction de modèles pour l'analyse et l'organisation de réseaux de flux : application à la gestion des effluents d'élevage à la Réunion*. PhD thesis, 2003.

[HGS08] A. Hélias, F. Guerrin, and J.-P. Steyer. Using timed automata and model-checking to simulate material flow in agricultural production systems - application to animal waste management. *Comput. Electron. Agric.*, 63(2) :183–192, October 2008.

[HNR68] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107, July 1968.

[HNSY94] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2) :193–244, 1994.

[JRLD07] J.-J. Jessen, J.-I. Rasmussen, K.-G. Larsen, and A. David. Guided controller synthesis for climate controller using uppaal tiga. In Jean-François Raskin and P. S. Thiagarajan, editors, *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, volume 4763 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2007.

[KW52] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3) :pp. 462–466, 1952.

[LC10] C. Largouët and M.-O. Cordier. Patrons de scenarios pour l'exploration qualitative d'un écosystème. *Actes de Reconnaissance des Formes et Intelligence Artificielle (RFIA'10)*, 2010.

[LCB+11] C. Largouët, M.-O. Cordier, Y.-M. Bozec, Y. Zhao, and G. Fontenelle. Use Of Timed Automata And Model-Checking To Explore Scenarios On Ecosystem Models. *Environmental Modelling and Software*, (30) :123–138, November 2011. On-line publication : 26 November 2011.

[LCF09] C. Largouët, M.-O. Cordier, and G. Fontenelle. Scenario templates to analyse qualitative ecosystem models. In R.D. Braddock, R.S. Anderssen, and L.T.H. Newham, editors, *International Congress on Modelling and Simulation (MODSIM'09)*, pages 2129–2135, Cairns, Australie, 2009.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1-2) :134–152, 1997.

[LTC+12] D. Leenhardt, O. Therond, M.-O. Cordier, C. Gascuel-Odoux, A. Reynaud, P. Durand, J.-E. Bergez, L. Clavel, V. Masson, and P. Moreau. A generic framework for scenario exercises using models applied to water-resource management. *Environmental Modelling and Software*, 37 :125–133, 2012.

[MCR11] R. Martin-Clouaire and J.-P. Rellier. A generic framework for simulating agricultural production systems. In D. Sauvant, J. Milgen, P. Faverdin, and N. Friggens, editors, *Modelling nutrient digestion and utilisation in farm animals*, pages 13–21. Wageningen Academic Publishers, 2011.

[Mfdrpldlm99] M. Merceron and Institut français de recherche pour l'exploitation de la mer. *Pollutions diffuses : du bassin versant au littoral : actes de colloques, Ploufragan (Saint-Brieuc), [23-24] Septembre 1999.* Actes de colloques - IFREMER. Ifremer, 1999.

[MMCD13] G. Martin, R. Martin-Clouaire, and M. Duru. Farming system design to feed the changing world. A review. *Agronomy for Sustainable Development*, 33(1) :131–149, January 2013.

[MMCRD11] G. Martin, R. Martin-Clouaire, J.-P. Rellier, and M. Duru. A simulation framework for the design of grassland-based beef-cattle farms. *Environ. Model. Softw.*, 26(4) :371–385, April 2011.

[MRM+08a] P.-T. Monteiro, D. Ropers, R. Mateescu, A.-T. Freitas, and H. De Jong. Temporal logic patterns for querying dynamic models of cellular interaction networks. (RR-6470), 2008.

[MRM+08b] P.-T. Monteiro, D. Ropers, R. Mateescu, A.-T. Freitas, and H. De Jong. Temporal Logic Patterns for Querying Qualitative Models of Genetic Regulatory Networks. In M. Ghallab, C.D. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *18th European Conference on Artificial Intelligence*, Patras, Greece, 2008. IOS Press.

[Mur02] J. D. Murray. *Mathematical Biology I : An Introduction*, volume 17 of *Interdisciplinary Applied Mathematics*. Springer New York, 2002.

[OD08] D.-L. Olson and D. Delen. *Advanced Data Mining Techniques*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[Pai11] C. Paillette. Ecomata appliqué à un écosystème halieutique de la mer du nord, 2011.

[RP97] J. Rickel and B.W. Porter. Automated modeling of complex systems to answer prediction questions. *Artificial Intelligence Journal*, 93 :201–260, 1997.

[RW87] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1) :206–230, January 1987.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 1989.

[SBA06] P. Salles, B. Bredeweg, and S. Araújo. Qualitative models about stream ecosystem recovery : Exploratory studies. *Ecological Modelling*, 194(1-3) :80–89, 2006.

[SMGOG+11] J. Salmon-Monviola, C. Gascuel-Odoux, F. Garcia, F. Tortrat, M.-O. Cordier, V. Masson, and R. Trépos. Simulating the effect of techniques and environmental constraints on the spatio-temporal distribution of herbicide applications and stream losses. *Agriculture, Ecosystems and Environment*, 140 :382–394, January 2011.

[TMC+12] R. Trépos, V. Masson, M.-O. Cordier, C. Gascuel-Odoux, and J. Salmon-Monviola. Mining simulation data by rule induction to determine critical source areas of stream water pollution by herbicides. *Computers and Electronics in Agriculture*, 86 :75–88, 2012.

[TN06] D. Tulllos and M. Neumann. A qualitative model for characterizing effects of anthropogenic activities on benthic communities. *Ecological Modeling*, 196 :209–220, 2006.

[Tré08] Cordier M.-O. Trépos, R. *Apprentissage symbolique à partir de données issues de simulation pour l'aide à la décision gestion d'un bassin versant pour une meilleure qualité de l'eau.* PhD thesis, 2008. Thèse doctorat Informatique Rennes 1 2008.

[TSAC+13] R. Trépos, A. Salleb-Aouissi, M.-O. Cordier, V. Masson, and C. Gascuel-Odoux. Building Actions From Classification Rules. *Knowledge and Information Systems (KAIS) journal*, 34(2) :267–298, 2013.

[VTZ+10] A. Veresi, S.L. Toffolatti, G. Zocchi, R. Guglielmann, and L. Ironi. A new approach to modelling the dynamics of oospore germination in plasmopara viticola. *European Journal of Plant Pathology*, 128(18) :113–126, 2010.

[Yov97] S. Yovine. Kronos : A verification tool for real-time systems (kronos user's manual release 2.2). *International Journal of Software Tools for Technology Transfer*, 1 :123–133, 1997.

[Yov98] S. Yovine. Model Checking timed automata. In *Embedded Systems*. LNCS Springer-Verlag, 1998.

[ZLC11] Y. Zhao, C. Largouët, and M.-O. Cordier. EcoMata : Un logiciel d'aide à la décision pour améliorer la gestion des écosystèmes. *Ingénierie des Systèmes d'Information (ISI)*, 16(3) :85–111, 2011.

Table des figures

Résumé

La modélisation dans le domaine de l'agro-écologie est importante car elle permet de mieux comprendre les interactions entre l'environnement et les activités humaines. Les travaux basés sur la simulation restent difficiles à utiliser dans un contexte d'aide à la décision. En particulier la recherche de stratégies optimales reste un grand défi. Nous proposons une approche qui consiste à représenter le système étudié dans un formalisme de système à événements discrets. Ceci permet de profiter de l'efficacité du model-checking et d'utiliser la synthèse de contrôleur pour générer automatiquement des stratégies optimales.

Nous présentons deux contributions dans cette thèse. La première contribution concerne le projet EcoMata. Cette modélisation qualitative en automates temporisés pour un réseau trophique marin de type proie-prédateur permet d'analyser l'écosystème à l'aide du model-checking. Nous avons amélioré la génération d'automates temporisés à partir des paramètres numériques. Nous avons aussi proposé une approche pour générer automatiquement des stratégies optimales de gestion de pêche. Dans la seconde contribution, nous proposons une modélisation hybride en automates temporisés pour la gestion de pâturage. Nous proposons quatre méthodes pour générer des stratégies optimales des activités de pâturage. La première méthode est appliquée à la recherche de stratégies optimales de mise au pâturage. Trois méthodes sont dédiées à la recherche de stratégies optimales de fertilisation. Une d'entre elles utilise la synthèse de contrôleur alors que les deux autres combinent la synthèse de contrôleur et l'apprentissage supervisé pour générer des stratégies génériques par type d'exploitation.

Abstract

Modeling in the domain of agro-ecology is important since it helps to better understand the interactions between the environment and the human activities. Some research based on simulation has been carried out during the recent years. However, not only these simulation tools are difficult to use by the non expert users, but also the high complexity of models makes interactive use almost impossible. We propose an approach in which the system is represented in a discrete event system formalism. This kind of representation takes advantage of the efficiency of model-checking and makes it possible to use controller synthesis to generate strategies.

This thesis contains two contributions. The first one concerns the project EcoMata which proposes a qualitative modeling for representing a marine prey-predator type food chain in timed automata. We have improved the efficiency of the algorithm of timed automata generation and developed a strategy synthesis method to generate best fishing management strategies. The second contribution, concerns a hybrid modeling which represents grazing activities in timed automata. This hybrid modeling combines a numerical grass model and a qualitative grazing model. We propose four methods to generate best grazing management strategy. One of these methods is applied to the movement of herd. The other three methods are applied to fertilization among which one of them uses controller synthesis on timed automata and the other two combine controller synthesis and machine learning to generate generic strategies per exploitation type.