Répondre aux questions "Que faire pour" par synthèse de contrôleur sur des automates temporisés - Application à la gestion de la pêche

Yulong Zhao, Marie-Odile Cordier, Christine Largouët

Répondre aux questions “Que faire pour” par synthèse de contrôleur sur des automates temporisés - Application à la gestion de la pêche

Yulong Zhao[1,3] Marie-Odile Cordier[1,3] Christine Largouët[2,3]

[1] Université de Rennes 1, Campus de Beaulieu, 35042 Rennes Cedex
[2] Agrocampus Ouest, 65 rue de Saint-Brieuc, 35042 Rennes Cedex
[3] IRISA, Campus de Beaulieu, 35042 Rennes Cedex

yulong.zhao@irisa.fr, cordier@irisa.fr, largouet@agrocampus-ouest.fr

Résumé

Nous montrons dans cet article comment répondre à des questions de type “Que faire pour éviter telle situation ?” (requête de sûreté) en nous appuyant sur une modélisation qualitative sous forme d’automates temporisés et en utilisant des outils de model-checking. Une approche exploitant la synthèse de contrôleur est comparée à une approche de type “Générer et tester”. L’application qui motive ce travail est celle de la gestion d’un écosystème marin et l’élaboration de politiques de gestion de pêche.

Mots Clef

Modélisation qualitative, automates temporisés, synthèse de contrôleur, model-checking

Abstract

This article presents two approaches to answer questions like “How to avoid this situation ?”(safety query). These approaches rely on a qualitative model in the form of timed automata and use model-checking tools. An approach exploiting controller synthesis is compared to a “generate and test” approach. The application that motivates this work is the management of a marine ecosystem and the evaluation of fishery management policies.

Keywords

Qualitative modelling, timed automata, controller synthesis, model-checking

1 Introduction

L’utilité des modèles de simulation pour explorer un domaine et aider à la décision est bien reconnue. Ceci est en particulier vrai dans le domaine de l’écologie où il est important de mieux comprendre les interactions souvent complexes entre différents acteurs, l’écosystème lui-même, l’environnement (climat par exemple), les humains en charge de l’exploitation de l’écosystème [16]. L’article [11] s’appuyait sur une modélisation qualitative de type automates temporisés, permettant de représenter de manière modulaire les différents modèles. Il proposait des patrons de scénarios facilitant l’expression, par un utilisateur non spécialiste, de requêtes dites prédictives de type “Peut-on atteindre telle situation ?”, “Quand atteindra-t-on telle situation ?” (atteignabilité), “Est-il possible (oui ou non) d’éviter telle situation ?” (sûreté). L’idée était de tirer parti des méthodes et outils de model-checking pour fournir une réponse à ces requêtes traduites dans la logique temporelle TCTL. Ce travail était illustré sur un écosystème marin décrivant l’évolution d’espèces de poissons soumis à des pressions de pêche et à des perturbations de l’environnement. Il a donné lieu à une expérimentation en grandeur nature pour explorer un écosystème corallien en Nouvelle-Calédonie (article accepté pour publication). Un logiciel EcoMata a été implémenté et est décrit dans [18].

Nous présentons dans cet article l’extension de ce travail 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 transitions contrôlables et transitions non contrôlables, de fournir des politiques répondant à un objectif donné, par exemple un objectif de sûreté [2]. Nous analysons les résultats et comparons avec une approche plus pragmatique, elle-aussi complète et utilisant le model-checking, dans une démarche de type “générer et tester”.

Après un bref rappel décrivant la modélisation et les principales options du travail décrit en [11] en section 2, nous présentons rapidement les automates de jeux et la synthèse de contrôleur en section 3. Nous montrons ensuite en section 4 comment est modélisé l’écosystème qui nous sert d’application illustrative. Nous présentons l’utilisation de l’approche synthèse de contrôleur et analysons les résultats obtenus avec le logiciel Uppaal-Tiga en section 5. Nous présentons ensuite l’approche de type “générer et tester” et les résultats obtenus en section 6 avant de conclure en ouvrant quelques perspectives.

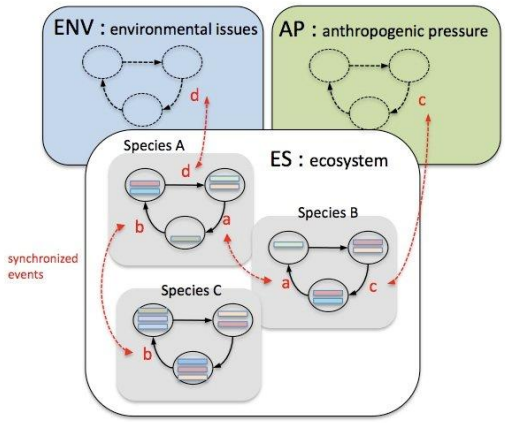

FIG. 1 – Modèle qualitatif de l'écosystème

2 Contexte de ce travail

Cette section fait un bref rappel de la modélisation qualitative que nous proposions en [11] pour représenter de manière générique un écosystème ainsi que le principe d'utilisation du model-checking pour répondre à des requêtes prédictives sur un tel modèle.

Le modèle que nous proposons se compose de trois sous-modèles : un réseau trophique *ES*, qui décrit l'évolution d'un écosystème de type proie-prédateur, les perturbations environnementales *ENV*, tels que l'occurrence d'événements environnementaux aléatoires ou périodiques (cyclone, le réchauffement) et les pressions anthropiques *AP*, décrivant les activités humaines agissant sur l'écosystème, telles que les activités de pêche (voir figure 1). Ces trois modèles sont représentés en utilisant le formalisme des automates temporisés [1].

Les états sont décrits de manière qualitative. Par exemple, les états de l'automate décrivant l'évolution des poissons sont décrits selon quatre valeurs qualitatives *Danger*, *Low*, *Normal* et *High* décrivant leur niveau de biomasse. Les états de l'automate décrivant la pression de pêche sur les espèces de poissons sont décrits par quatre valeurs qualitatives *Stopped*, *Low*, *Normal* et *High*.

Les transitions d'un automate temporisé sont étiquetées par des contraintes temporelles. Il existe deux types de contraintes temporelles, les contraintes sur les états appellées *invariant* (on peut entrer et rester dans un état tant que l'invariant est vrai) et les contraintes sur les transitions appellées *garde* (une transition ne peut être déclenchée que quand la garde est vraie). On peut en particulier exprimer ainsi la durée nécessaire du passage entre deux états qualitatifs stables. Les interactions entre modèles, en général complexes, s'expriment par l'existence d'événements partagés forçant leur synchronisation grâce à une horloge commune.

Les patrons de scénarios définis en [11] correspondent aux requêtes prédictives les plus fréquemment posées par les utilisateurs. Ces patrons sont basés sur la notion de *situation* de l'écosystème. Une *situation* est l'état global de l'écosystème représenté par des valeurs qualitatives de la biomasse pour chacune des espèces. Ces patrons de scénarios sont traduits automatiquement en TCTL (*Timed Computation Tree Logic*), langage de logique temporelle. A l'aide de la technique du *model-checking*, nous pouvons explorer le modèle global et fournir des résultats (booléens ou dates). Ceci nous permet d'évaluer l'impact d'actions anthropiques, par exemple l'impact de politiques de pêche sur le stock d'espèces de poissons, mais ne permet pas de construire des politiques répondant à certains critères, par exemple construire la politique de gestion de pêche évitant que le niveau de biomasse des espèces de poisson ne passe au niveau *Danger*.

L'objectif du travail présenté ici est d'étendre le travail présenté en [11] et permettre d'exprimer et de répondre à des requêtes nécessitant la construction de politiques (ensemble de règles de comportement) garantissant d'éviter certaines situations non souhaitables (politique de sûreté).

3 Synthèse de contrôleur

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

3.1 Automate temporisé de jeux

Un *automate temporisé* [1] est un n-uplet $\mathcal{A} = (Q, q_0, \mathcal{X}, T, \mathcal{G})$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état de départ ;
- $\mathcal{X}$ est un ensemble fini d'horloges ;
- $T \subseteq Q \times Q$ est un ensemble fini de transitions ;
- $\mathcal{G} \subseteq T \times \phi$ où $\phi ::= \{\bigwedge x\ op\ c | x \in \mathcal{X},\ c \in \mathbb{N},\ op \in \{<, \leq, =, \geq, >\}\}$ est un ensemble de conditions (sur les horloges) de déclenchement de transitions.

Un *automate temporisé de jeux* est une extension d'un automate temporisé dans laquelle nous séparons les transitions en deux catégories : les transitions contrôlables T^C et les transitions non contrôlables T^E [5]. Notons $\mathcal{G}^C$ et $\mathcal{G}^E$ les conditions de déclenchement qui y sont associées. Un *automate temporisé de jeux* est un n-uplet $\mathcal{A} = (Q, q_0, \mathcal{X}, T^E, T^C, \mathcal{G}^E, \mathcal{G}^C)$

3.2 Algorithme de synthèse

On voit qu'il est possible de renforçer les conditions de déclenchement des transitions contrôlables, en remplaçant $\mathcal{G}^C$ par $\mathcal{G}^C_* \subseteq \mathcal{G}^C$. On peut ainsi *contrôler* le comportement de l'automate afin de rendre certains états inatteignables. La condition que l'on ajoute à une transition est appelée une politique. Le contrôleur est l'ensemble de ces politiques [4] [1].

Le problème de la synthèse de contrôleur est défini comme suit : étant donné une propriété φ, existe-il un contrôleur

[1] On peut aussi définir le contrôleur comme l'automate dont la synchonisation avec l'automate de jeux correspondra à l'ajout des conditions de déclenchement sur les transitions contrôlables.

C, et donc un ensemble de conditions de déclenchement des transitions contrôlables $\mathcal{G}_*^C$, tel que toute trajectoire[2] de l'automate $\mathcal{A}_* = (Q, q_0, \mathcal{X}, T^E, T^C, \mathcal{G}^E, \mathcal{G}_*^C)$ satisfait la propriété φ ? 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 de la sûreté, qui est celui qui nous intéresse ici, la propriété φ décrit les situations que toute trajectoire doit éviter.

Etant donné un automate temporisé de jeux $\mathcal{A} = (Q, q_0, \mathcal{X}, T^E, T^C, \mathcal{G}^E, \mathcal{G}^C)$ et une propriété de sûreté φ, la synthèse de contrôleur commence par déterminer le sous-ensemble d'états $Q_* \subseteq Q$ où la propriété φ est vérifiée. Il cherche ensuite un sous-ensemble $\mathcal{G}_*^C \subseteq \mathcal{G}^C$ tel que $\mathcal{A}_* = (Q_*, q_0, \mathcal{X}, T^E, T^C, \mathcal{G}^E, \mathcal{G}_*^C)$ est non bloquant[3] et les états de toute trajectoire appartiennent exclusivement à Q_*. Si le sous-ensemble $\mathcal{G}_*^C$ n'est pas vide, on peut conclure qu'un tel contrôleur existe. Les conditions de $\mathcal{G}_*^C$ permettent de construire le contrôleur. Celui-ci n'est pas forcément unique.

4 Modélisation et méta-modèle

Nous montrons dans cette section comment nous avons étendu la modélisation existante à base d'automates temporisés 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. Nous illustrons le tout sur notre application.

4.1 Automates temporisés de jeux pour modéliser l'écosystème

Comme expliqué en section 2, le modèle global décrit un réseau trophique décrivant l'évolution de l'écosystème (espèces de poissons), les perturbations environnementales (non considérées dans cet article) et anthropiques (pressions de pêche). Nous nous intéressons en particulier aux pressions anthropiques qui sont les seules transitions contrôlables de notre modèle. Nous avons donc :

- des automates non contrôlables (leurs transitions ne sont pas contrôlables) : ce sont les automates représentant l'écosystème et les perturbations environnementales. Nous y ajoutons les pressions de pêches 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 anthropiques dont la politique est à synthétiser, pour nous ce seront des pressions de pêches.

Nous obtenons ainsi un ensemble d'automates temporisés de jeux (voir section 3).

[2]Une trajectoire d'un automate est une suite d'états et de transitions où le premier état est l'état de départ $q0$ et où toutes les transitions satisfont les conditions de déclenchement.

[3]Un automate est non bloquant si, pour tout état, il existe au moins une transition satisfaisant les conditions de déclenchement.

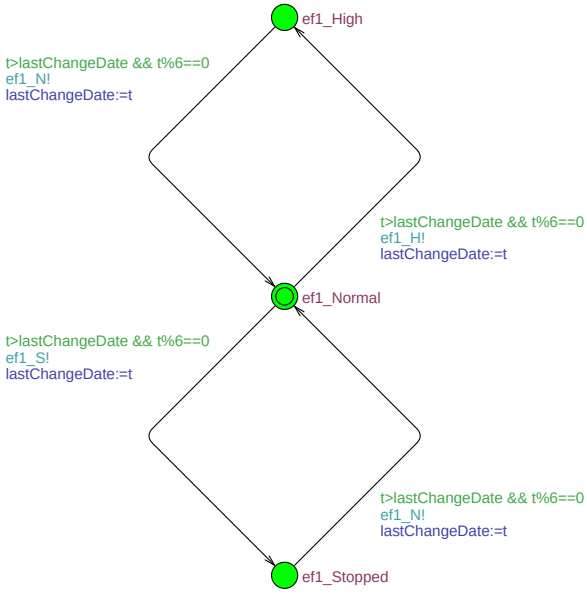

FIG. 2 – Un exemple de l'automate de pêche

4.2 Structure de l'automate de pêche et méta-modèle

Partant de la modélisation des pressions anthropiques, et en particulier des pressions de pêche décrite dans [11], nous avons identifié les contraintes que les pressions de pêche, et donc les automates les décrivant, doivent respecter pour être réalistes.

- 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 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 anthropiques doivent être satisfaites par tous les comportements que propose le contrôleur. Nous avons choisi de les exprimer formellement par un méta-modèle. Les automates de jeux devront être des instances de ce méta-modèle.

La figure 2 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 que nous avons utilisé (voir section 5). Les trois cercles représentent des états qualitatifs de la force de pêche. Une horloge t et une variable locale *last-*

ChangeDate 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 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 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.

5 Expérimentations et résultats

5.1 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 [5]. 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 politiques (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 [12]. Les automates sont obtenus en ajoutant une étiquette de contrôlabilité sur les transitions 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.
Soit un automate $\mathcal{A} = (Q, q_0, \mathcal{X}, T^E, T^C, \mathcal{G}^E, \mathcal{G}^C)$ et l'ensemble des automates $\mathcal{A}_* = (Q, q_0, \mathcal{X}, T^E, T^C, \mathcal{G}^E, \mathcal{G}^C_*)$ évitant la propriété φ. Soit l'ensemble $\mathcal{G}_* = \{\mathcal{G}^C_*\}$, deux types de contrôleurs peuvent être intéressants :

– le contrôleur complet $\mathcal{G}^C_{*max} \in \mathcal{G}_*$ tel que pour tout $\mathcal{G}^C_* \in \mathcal{G}_*$, on a $\mathcal{G}^C_* \subseteq \mathcal{G}^C_{*max}$;
– les contrôleurs minimals $\mathcal{G}^C_{*min} \in \mathcal{G}_*$ tel qu'il n'existe aucun $\mathcal{G}^C_* \in \mathcal{G}_*$ avec $\mathcal{G}^C_* \subset \mathcal{G}^C_{*min}$.

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 stratégies solutions ; soit un des contrôleurs minimals, choisi aléatoirement, qui correspond à une stratégie solution.

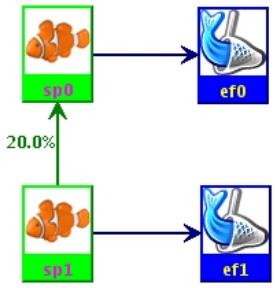

FIG. 3 – Ce réseau trophique est composé de deux espèces de poisson. *sp0* est soumis à la force de pêche *ef0* et *sp1* à la force de pêche *ef1*. Le pourcentage sur la flèche de *sp0* vers *sp1* indique que 20% de nourriture de *sp0* vient de la biomasse de *sp1*.

Rappelons qu'un contrôleur est décrit par un ensemble de politiques. Une politique est une condition de déclenchement, associée à une transition, décrivant l'état du système dans lequel il faut déclencher la transition. Cet état global est exprimé par l'ensemble des états des automates composant le modèle global, ainsi que les valeurs des variables et des valeurs (ou intervalles) des horloges.

5.2 Résultats expérimentaux

Nous avons choisi dans un premier temps d'utiliser un réseau trophique de taille réduite pour les premières expérimentations (cf. figure 3). Pour les politiques de pêche, nous nous restreignons à trois niveaux qualitatifs (*Stopped*, *Normal* et *High*) et nous imposons un intervalle multiple de 6 unités de temps (6 mois) entre deux changements d'état. Nous supposons de plus 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 *Danger* s'exprime en logique TCTL par la formule :

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

Nous demandons à UPPAAL-TIGA de nous fournir un des contrôleurs minimals s'il en existe un. Une politique proposée et interprétée en langage naturel est : «Quand les espèces de poissons *sp0* et *sp1* sont dans l'état où le niveau de biomasse est *Normal*, à l'unité de temps 0, augmenter le niveau de pêche de *ef1* à *High*».
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 qui nous motivent.

– 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 ;

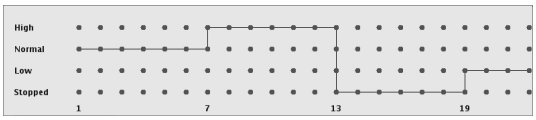

FIG. 4 – Planning de pêche sous forme de chronogramme

– 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 la transition à déclencher. 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 conditions de déclenchement convient pour un contrôleur automatique en ligne, 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 que nous avons explorée afin de trouver une solution n'ayant pas les inconvénients listés ci-dessus.

6 Synthèse par une approche "Générer et tester"

6.1 Motivations

Les difficultés exposées ci-dessus nous 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. Nous avons décidé de construire des politiques sous cette forme, tout en utilisant le simulateur pour qu'elles soient cohérentes avec le niveau de biomasse prédit.

Un chronogramme est exprimé sous la même forme que dans le logiciel ECOMATA [18] (voir figure 4) et 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 [18] 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.

Nous avons ainsi décidé de nous appuyer sur ce "vérificateur" pour construire et tester des politiques de pêche en suivant une approche "Générer et tester".

6.2 Génération et test des politiques

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

– Énumérer toutes les politiques possibles respectant les contraintes fournies en entrée telles que le nombre de niveaux de pressions de pêche considérés, la durée d'une politique (nombre d'unité de temps du chronogramme), l'intervalle entre deux changements.
– Trier les politiques générés selon une fonction d'évaluation qui ne dépend que de la nature de la politique. Par exemple, cette fonction favorise le moindre nombre de changements, ou une quantité maximale de biomasse pêchée, ou une qualité de l'écosystème en termes de survie des espèces etc.
– Tester les politiques dans l'ordre afin de déterminer si elles répondent aux critères exprimés dans la requête. On arrête le test dès qu'une politique satisfaisant aux critères de la requête est trouvée.

6.3 Expérimentations et résultats

Pour pouvoir comparer cette méthode et la méthode proposée dans la section 5, nous utilisons le même écosystème (cf. figure 3). Nous générons des politiques pour des pressions 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. Nous accordons un poids 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 *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 *Danger*
– éviter que l'espèce *sp1* soit dans l'état *Danger* ou dans l'état *Low*

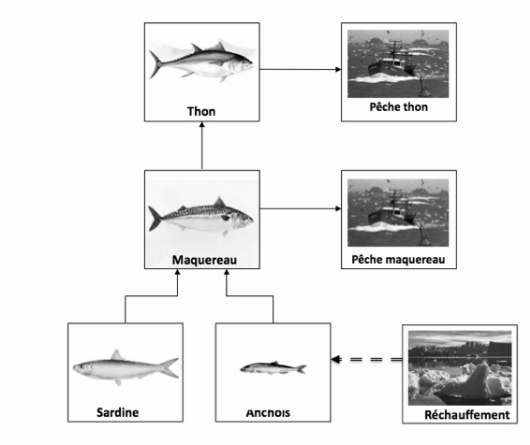

FIG. 5 – Réseau trophique thon et maquereau

– éviter que l'espèce *sp1* soit dans l'état *Danger* ou que l'espèce *sp0* soit dans l'état *Danger*
– éviter que l'espèce *sp1* soit dans l'état *Danger* et que l'espèce *sp0* soit dans l'état *Danger*

Nous avons fait nos expérimentations en faisant varier la politique de *ef0* (supposée figée) pour tester la performance de l'algorithme. Les tables 1, 2 et 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.

Enfin, nous avons fait des expérimentations avec un modèle plus complexe (cf. figure 5) sur un écosystème différent. 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. Nous recherchons 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, nous avons ajouté un mécanisme de minutage 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.

7 Conclusion

Cet article présente une extension au travail présenté dans [11] afin de permettre de traiter des requêtes de type "Que faire pour éviter une 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'augmenter le logiciel ECOMATA [18] pour permettre à un utilisateur d'explorer un modèle qualitatif complexe et pour l'aider à prendre des décisions. Dans notre cas, nous sommes motivés par une application portant sur un écosystème marin, où l'évolution de plusieurs espèces de poissons dépend de leurs interactions de type proie-prédateur, mais dépend aussi des conditions environnementales (climatiques en particulier) et de la pression anthropique que font subir les pêcheurs. Il est donc important de pouvoir déterminer les politiques de gestion de pêche permettant d'éviter des situations non souhaitables.

La modélisation qualitative a été reconnue comme tout à fait pertinente [15], en particulier lorsqu'il s'agit d'explorer des domaines complexes tels que la physiologie végétale [14], la pollution de l'eau [7, 8, 16] et les écosystèmes [9]. Certains travaux s'intéressent plus directement aux interactions entre activités anthropiques et biosystème tel que [17] et l'utilisation de modèles qualitatifs dans le contexte d'aide à la décision a été récemment recommandée dans [19]. Les auteurs de [3] insistent sur le fait que les modèles doivent être construits par un échange interactif entre l'utilisateur et l'outil permettant les simulations. Nous nous situons tout à fait dans ce contexte. Un travail assez proche est celui décrit dans [13] dans le domaine des réseaux de régulation génétiques où est proposé un menu regroupant les questions les plus fréquemment posées par un utilisateur. Cependant le formalisme retenu est celui des automates, sans prise en compte des contraintes temporelles et de plus, il ne traite que des requêtes prédictives et ne construit pas de politiques répondant à un objectif comme nous le faisons dans cet article.

En ce qui concerne l'approche de model-checking, elle a été utilisé avec succès pour la vérification automatique de systèmes complexes [6, 10]. Les travaux les plus proches de notre problématique sont principalement théoriques et concernent l'étude d'algorithmes efficaces pour l'analyse en ligne d'automates de jeux [5].

Nous présentons tout d'abord une approche s'appuyant sur la synthèse de contrôleur, issue du model-checking, et en analysons les résultats. Nous présentons ensuite une autre approche, de type "Générer et tester", motivée par les difficultés rencontrées avec la précédente approche. Cette seconde approche s'appuie plus directement sur l'outil ECOMATA qui est utilisé dans la phase "tester". Nous en présentons les principes, l'algorithme et analysons les résultats obtenus.

L'approche directe par synthèse de contrôleur est séduisante et générique. Elle fournit un ensemble de politiques qui permet, quelque soit l'état atteingnable 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). L'approche de type "Générer et tester" est plus ad-hoc et utilise directement la plate-forme ECOMATA que nous avons développée, mais le mode d'expression des stratégies s'inspire de celui couramment employé par un gestionnaire d'écosystèmes, à savoir la planification d'actions anthropiques (ici la 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. Après avoir expérimenté et comparé ces deux approches, il apparaît ainsi que la seconde approche répond mieux au problème posé. Nous prévoyons de rendre disponible ces nouvelles capacités du logiciel (voir `http://oban.agrocampus-ouest.fr/ecomata`), qui permettent à l'utilisateur d'explorer le modèle par des requêtes dites proactives, et fournissent une stratégie, en plus des requêtes prédictives qui vérifient seulement l'atteignabilité. Nous voulons ensuite étendre notre étude en ne nous limitant plus aux requêtes proactives de type sûreté "Que faire pour éviter telle situation" mais en élargissant aux requêtes de type "Que faire pour atteindre tel objectif ?".

Références

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

[2] E. Asarin and O. Maler and A. Pnueli and J. Sifakis *Controller Synthesis For Timed Automata*, 1998

[3] J. M. Attonaty and M. H. Chatelin and F. Garcia Interactive simulation modeling in farm decision making *Computers and Electronics in Agriculture*, Vol. 22, N° 2-3, pp. 157-170, 1999

[4] V. Aymeric Synthèse de contrôleurs et stratégies gagnantes dans les jeux de parité *Actes du 3ème Colloque Francophone sur la Modélisation des Systèmes réactifs (MSR'01)*, Hermès, pp 87-98, 2001

[5] F. Cassez and A. David and E. Fleury and K.-G. Larsen and D. Lime Efficient on-the-fly algorithms for the analysis of timed games *In Proceedings of CONCUR 05, LNCS 3653* Springer, pp. 66-88, 2005

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

[7] M.-O. Cordier and F. Garcia and C. Gascuel-Odoux and V. Masson and J. Salmon-Monviola and 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 Proceedings of MODSIM'05 (International Congress on Modelling and Simulation)*, 2005

[8] F. Guerrin Qualitative reasoning about an ecological process : interpretation in hydroecology *Ecological Modelling*, Vol. 59, N° 2, pp. 165-201, 1991

[9] F. Guerrin and J. Dumas Knowledge representation and qualitative simulation of salmon redd functioning. Part I : qualitative modeling and simulation *Biosystems*, N° 2, pp. 75-84, 2001

[10] T. Henzinger and X6. Nicollin and J. Sifakis and S. Yovine Symbolic model checking for real-time systems *Information and Computation*, Vol. 111, N° 2, pp. 193-244, 1994

[11] 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

[12] K.G. Larsen and P. Pettersson and W. Yi UPPAAL in a Nutshell *Journal of Software Tools for Technology Transfer*, Vol. 1, N° 1-2, pp. 134-152, 1997

[13] P.-T. Monteiro and D. Ropers and R. Mateescu and A.-T. Freitas and H. de Jong Temporal logic patterns for querying dynamic models of cellular interaction networks *ECCB*, pp. 227-233, 2008

[14] J. Rickel and B.W. Porter Automated Modeling of Complex Systems to Answer Prediction Questions *Artificial Intelligence Journal*, Vol. 93, pp. 201-260, 1997

[15] J. Rykiel Artificial Intelligence and Expert Systems in Ecology and Natural Resources Management *Ecological Modelling*, Vol. 46, N° 1-2, pp. 3-8, 1989

[16] P. Salles and B. Bredeweg and S. Araújo Qualitative models about stream ecosystem recovery : Exploratory studies *Ecological Modelling*, Vol. 194, N° 1-3, pp. 80-89, 2006

[17] D. Tulllos and M. Neumann A Qualitative Model for Characterizing Effects of Anthropogenic Activities on Benthic Communities *Ecological Modeling*, Vol. 196, pp. 209-220, 2006

[18] Y. Zhao and 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*, Vol. 16, N° 3, pp 85-111, 2011

[19] A. Zitek and S. Schmutz and S. Preis and P. Salles and B. Bredeweg and S. Muhar Evaluating the potential of qualitative reasoning models to contribute to sustainable catchment management *Ecological Informatics*, Vol. 4, N° 5-6, pp. 381-395, 2009

Politique de *ef0* : 0 0 3 3 3 3				
Requête	Nb. politiques testées	Temps total d'exécution	Temps moyen	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. 1 – Résultats de synthèse par l'approche "génener 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	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. 2 – Résultats de synthèse par l'approche "génener 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	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. 3 – Résultats de synthèse par l'approche "génener 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	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 – Résultats de synthèse par l'approche "génener et tester" pour la pression de pêche ef1 sur l'espèce $sp1$, pour une durée de 48 unités de temps