Algorithme d'approximation du noyau de viabilité avec procédure de classification

Wei Wei, Isabelle Alvarez, Sophie Martin

Algorithme d'approximation du noyau de viabilité avec procédure de classification

W. Wei[1] I. Alvarez[1,2] S. Martin[1]

[1] Laboratoire d'Ingénierie pour les Systèmes Complexes (LISC) Cemagref
24 Avevue des landais B.P. 50085 - 63172 Aubière Cedex
[2] Laboratoire d'Informatique de Paris 6 (LIP6), CNRS - Université Pierre et Marie Curie
4 place Jussieu - 75252 PARIS cedex 05

wei.wei@cemagref.fr
isabelle.alvarez@cemagref.fr
sophie.martin@cemagref.fr

Résumé

La théorie de la viabilité propose des concepts et méthodes pour contrôler un système dynamique afin de le maintenir dans un ensemble de contraintes de viabilité. Intégrer les modèles de la viabilité dans un outil d'aide à la gestion participative est une nouvelle application de la théorie de la viabilité. Dans ce contexte, nous proposons un algorithme d'approximation du noyau de viabilité qui se concentre sur la réduction du temps de calcul. Cet algorithme utilise une procédure de classification du type "plus proche voisin". L'algorithme satisfait les conditions de convergence, et est facile à manipuler en utilisant une interface graphique utilisateur. Nous comparons les résultats obtenus pour des problèmes de viabilité pour lesquels les noyaux théoriques ont été calculés. La fidélité de cet algorithme est toujours supérieure à 90 pourcents. Le temps de calcul est très intéressant pour les outils d'aide à la décision.

Mots clés

Théorie de viabilité, Système dynamique, Gestion participative, Noyau de viabilité, Aide à la Décision.

Abstract

Viability theory proposes concepts and tools to control a dynamical system such that it can remain inside a viability constraint set. Integrating viability models in a tool support for participatory management is a new application of the viability theory. In this context, we propose an algorithm for viability kernel approximation, which focuses on improving the computing time. This algorithm uses the classification procedure based on the method of Nearest Neighbor Search. The algorithm satisfies the conditions of convergence, and is easy to use with a graphical user interface. We compare its results for models for which the exact viability kernels have been determined. The accuracy of this algorithm is always greater than 90 percent. The computing time is very interesting for decision support tools.

Keywords

Viability theory, Dynamical system, Participatory management, Viability Kernel, Decision Support.

1 Introduction

La théorie mathématique de la viabilité initiée dans les années 1990 par Jean-Pierre Aubin [1], étudie la compatibilité entre des dynamiques et des contraintes. Le concept essentiel est celui de noyau de viabilité qui rassemble l'ensemble des états du système à partir desquels il existe au moins une fonction de contrôle qui assure le respect des contraintes indéfiniment. Plusieurs algorithmes ont été développés pour approcher le noyau de viabilité : l'algorithme de Patrick Saint-Pierre [2] utilise une discrétisation de l'espace des états sur une grille régulière. Kviar, l'algorithme de Guillaume Deffuant *et al.* [3] utilise des Support Vector Machines. Coquelin *et al.* [4] proposent une approche de programmation dynamique qui ne permet pas de gain en espace mémoire mais en temps de calcul.

Les domaines dans lesquels la théorie de viabilité est utilisée sont très variés (économie, sciences cognitives, théorie des jeux, biologie, automatique ..), citons en bio-économie et en écologie [5]-[7]. Les jeux sérieux sont une approche nouvelle et efficace pour explorer et tester des possibilités d'évolutions dans un contexte réaliste mais sans coût ni risque [8]. En effet, ces jeux sont un substitut intéressant à l'expérience directe dans le monde réel ou sur des infrastructures réelles, qui peuvent permettre des apprentissages relativement rapides et sans danger [9].
Dans le domaine de la gestion de la biodiversité [10],

un défi important concerne la gestion d'aires protégées comme les parcs nationaux, qui sont soumis à de nombreuses pressions sur l'utilisation et l'accès aux ressources ce qui engendre souvent des conflits. Le projet SimParc [13] appartient au domaine de l'aide à la gestion participative et a pour but d'aider les différents acteurs à créer une compréhension collective des conflits et à négocier des stratégies pour y faire face. Le gestionnaire du parc agit comme un arbitre dans le jeu, il prend une décision finale sur les types de conservation à attribuer à chaque unité de paysage. Les autres acteurs (joueurs) font des propositions. Il y a également des agents artificiels qui peuvent remplacer certains acteurs et des agents experts capables de renseigner les autres joueurs sur la viabilité des solutions proposées (en terme de préservation des espèces menacées, par exemple) [12].
Or, dans le contexte de l'outil d'aide à la gestion participative SimParc, l'agent expert viabilité doit pouvoir fournir aux joueurs rapidement des résultats sur la viabilité des solutions proposées. Malheureusement, l'algorithme de Patrick Saint-Pierre [2] est difficile d'emploi dans le cadre d'un jeu sérieux où les modifications du modèle doivent être prises en compte très rapidement alors qu'il n'y a pas encore d'interface pour gérer les modifications. Kviar [3] dispose d'une telle interface mais l'algorithme est très lent. L'approximation du noyau de viabilité de Coquelin *et al.* [4] est l'ensemble des points de la grille pour lesquels la fonction valeur prend des valeurs inférieures à un seuil dont la détermination n'est pas évidente. Par conséquent, nous proposons un nouvel algorithme d'approximation du noyau de viabilité muni d'une interface et utilisant une procédure de classification conçue pour accélérer le temps de calcul. Les contraintes, la plage des contrôles possibles sont proposés par les utilisateurs et font l'objet de discussions, ainsi que le choix des contrôles à mettre en oeuvre de manière effective.

Ce papier sera divisé en trois parties. Tout d'abord, nous rappelons l'algorithme de calcul d'une approximation du noyau de viabilité à l'aide d'une procédure de classification et les conditions de sa convergence. Ensuite, nous allons décrire l'ensemble des fonctions de classification que nous proposons d'utiliser et montrer qu'elles respectent bien les conditions de convergence précédentes. Enfin, nous allons présenter une interface graphique utilisateur et comparer les noyaux obtenus pour des modèles dont le noyau théorique peut être calculé.

2 L'approximation du noyau de viabilité discret avec une procédure de classification

2.1 Contexte et notations

Nous considérons un système dynamique défini par son état $\vec{x}(t) \in X \subset R^N$ et nous supposons que son évolution peut être influencée par un contrôle $\vec{u}(t)$:

$$\begin{cases} \vec{x}'(t) & = & \varphi(\vec{x}(t), \vec{u}(t)) \\ \vec{u}(t) & \in & U(\vec{x}(t)) \end{cases} \quad (1)$$

L'ensemble des contrôles admissibles peut dépendre de l'état du système, $\vec{u}(t)$ est choisi dans un sous-ensemble $U(\vec{x}(t)) \subset R^q$.

L'ensemble des contraintes de viabilité est un sous-ensemble de X noté K. Le noyau de viabilité de K pour la dynamique ϕ définie par (1) est :

$Viab(K) = \{\vec{x} \in K \mid \exists \vec{u}(.) \mid \vec{x}(t) \in K \ \forall t \in [0, +\infty[\}$.
$Viab(K)$ rassemble les états dans X à partir desquels on trouve toujours une suite de contrôles $\vec{u}$ qui permette au système de rester dans l'ensemble des contraintes K.
Considérons un intervalle de temps *dt*, le système dynamique discret en temps associé à (1) est défini par la correspondance $G : X \mapsto X$ (qui associe à $\vec{x}$ l'ensemble de ses successeurs) :

$$G(\vec{x}) \quad = \quad \{\vec{x} + \varphi(\vec{x}, \vec{u})dt, \vec{u} \in U(\vec{x})\}. \quad (2)$$

Nous supposons que G est une correspondance μ-Lipschitz à images fermées [1], c'est à dire que les images de deux points quelconques ne peuvent pas être éloignées de manière arbitraire, mais au contraire restent dans un voisinage l'une de l'autre proportionel à la distance des points de départ.
Notre objectif est d'approcher $Viab_G(K)$ qui est le noyau de viabilité de la dynamique discrète (2). D'après les théorèmes de viabilité [1], $Viab_G(K)$ est le plus grand sous-ensemble E de K tel que :

$$\forall \vec{x} \in E, G(\vec{x}) \bigcap E \neq \emptyset \quad (3)$$

L'algorithme utilisant des fonctions de classification décrit dans [3] est le suivant :

Soit une grille, K_h, telle que :

$$\forall \vec{x} \in K, \exists \vec{x}_h \in K_h, \text{ tel que } \|\bar{x} - \vec{x}_h\| \leq \beta(h) \quad (4)$$

avec $\beta(h) \longrightarrow 0$ quand $h \longrightarrow 0$. Une telle grille existe puisque K est compact.

De plus,
- l est une procédure d'apprentissage qui associe à un ensemble S de paires $(\vec{x}_i, e_i) \in \mathrm{K} \times \{-1, 1\}$, une fonction de classification $l_S(\vec{x}) : K \longrightarrow \{-1, 1\}$.
- $d(E, F)$, la distance entre deux sous-ensembles E et F.
- $E \backslash F$, l'ensemble complémentaire de F dans E lorsque $F \subset E$.

[1] $\forall \vec{x},\ \vec{x}' \in K, G(\vec{x}') \subset G(\vec{x}) + \mu||\vec{x} - \vec{x}'||B$ où B est la boule unité et $\forall \vec{x} \in K$, l'ensemble $G(\vec{x})$ est fermé

2.2 Les étapes de l'algorithme

A chaque étape, sont définis un ensemble discret $K_h^n \subset K_h^{n-1} \subset K_h$, et un ensemble continu, noté $L(K_h^n)$, qui est une généralisation de cet ensemble discret, et qui constitue l'approximation courante du noyau de viabilité :

1. Initialisation : $K_h^0 := K_h$ et L(K_h^0) :=K.
2. Récurrence :
   - Définition de l'ensemble discret K_h^{n+1} à partir de K_h^n et L(K_h^n) :

     K_h^{n+1}=$\{\vec{x}_h \in K_h^n \mid d(G(\vec{x}_h), L(K_h^n)) \leq \mu\beta(h)\}$
   - si $K_h^{n+1} \neq K_h^n$, utiliser la procédure d'apprentissage *I* avec les points $\vec{x}_h$ de la grille K_h, avec l'étiquette +1 si $\vec{x}_h \in K_h^{n+1}$ et l'étiquette -1 sinon. Soit l_h^{n+1} la fonction de classification obtenue de K dans $\{-1, 1\}$, $L(K_h^{n+1})$ est défini ainsi :

$$L(K_h^{n+1}) \quad = \quad \left\{\vec{x} \in K, \mid l_h^{n+1}(\vec{x}) = +1\right\} \tag{5}$$

   - sinon, arrêt et retourner $L(K_h^n)$.

2.3 Le théorème de convergence de l'algorithme

La démonstration de ce théorème est donnée dans [3].

Theorem 1 *S'il existe un réel $\lambda \geq 1$ tel que, pour toutes les itérations n, l'approximation $L(K_h^n)$ satisfait les conditions suivantes :*

$$\forall \vec{x} \in L(K_h^n) \qquad d(\vec{x}, K_h^n) \leq \lambda\beta(h) \tag{6}$$

$$\forall \vec{x} \in K \backslash L(K_h^n) \qquad d(\vec{x}, K_h \backslash K_h^n) \leq \beta(h) \tag{7}$$

alors, l'algorithme d'approximation du noyau de viabilité fournit un résultat qui converge vers le noyau de viabilité exact lorsque le pas de la grille h tend vers 0.

Ces conditions signifient que tout point de $L(K_h^n)$ doit être près d'un point de K_h^n et que tout point de $K \backslash L(K_h^n)$ doit être près d'un point de $K_h \backslash K_h^n$.

3 Choix de la procédure de classification

Malgré leur utilité pour le calcul de noyaux de viabilité, les SVMs classiques ne vérifient pas toujours les conditions du théorème 1. Nous proposons ici d'utiliser des fonctions de type "plus proche voisin" qui vérifient les conditions du théorème.

3.1 Les fonctions de classification utilisées

Nous nous plaçons dans le cas d'une grille K_h régulière de pas h. Ainsi, $\beta(h) = \sqrt{N}h$.
Nous notons y_0 la fonction "plus proche voisin" :

$$\begin{array}{rcl} y_0 & : & K \rightarrow K_h \\ & & \vec{x} \rightarrow y_0(\vec{x}) = \text{argmin}(d(\vec{x}, K_h)) \end{array} \tag{8}$$

C'est le point le plus proche de $\vec{x}$ dans K_h, et nous avons $y_0(\vec{x}) \in K_h$ et $d(\vec{x}, y_0(\vec{x})) \in \left[0, \sqrt{N}h/2\right]$.

Nous posons également :

$$\begin{array}{l} y_i(\vec{x}) \subset K_h \\ := \{\vec{y} \in K_h \ \ , d(y_0(\vec{x}), \vec{y}) = h, d(\vec{x}, \vec{y}) \leq \sqrt{N+3}h/2\} \end{array}$$

Soit $S \subset K_h$. Si les points de $S \subset K_h$ sont associés à l'étiquette +1 et ceux de $K_h \backslash S$ à l'étiquette -1, nous proposons d'associer la fonction de classification suivante :

$$\begin{array}{lcl} l_S^*(\vec{x}) & = & +1 \\ \text{si } y_0(\vec{x}) \in S & \text{et} & \forall \vec{y} \in y_i(\vec{x}) | \vec{y} \notin S, \ \ d(\vec{x}, \vec{y}) > \alpha h \\ l_S^*(\vec{x}) & = & -1 \\ \text{sinon,} & & \end{array} \tag{9}$$

avec $\alpha \in [0, 1[$.

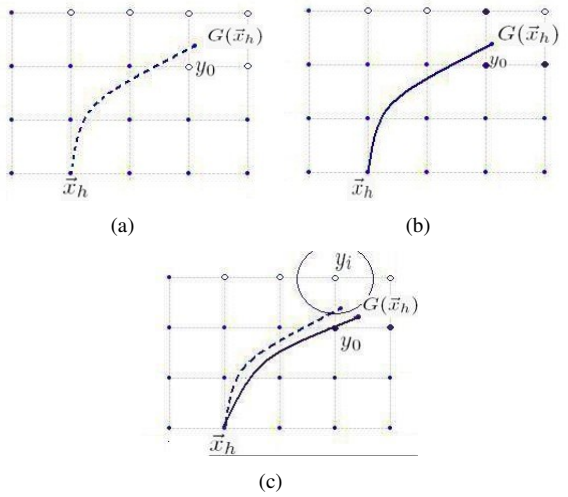


FIG. 1: Les points noirs appartiennent à K_h^n, les points blancs à son complémentaire dans K_h. Cas 1(a), $\vec{x}_h \notin K_h^{n+1}$ car $y_0(G(\vec{x}_h)) \notin K_h^n$. Cas 1(b), $\vec{x}_h \in K_h^{n+1}$ car $y_0(G(\vec{x}_h)) \in K_h^n$ et $\forall \vec{y} \in y_i(G(\vec{x}_h))$, $\vec{y} \in K_h^n$. Cas 1(c), $y_0(G(\vec{x}_h)) \in K_h^n$ mais $\exists \vec{y} \in y_i(G(\vec{x}_h))$, $\vec{y} \notin K_h^n$. Donc pour déterminer si $\vec{x} \in K_h^{n+1}$, il faut vérifer que $d(G(\vec{x}_h), \vec{y}) > \alpha h$.

3.2 Démonstration de la convergence

Dans cette section, nous allons vérifier que l^* vérifie bien les conditions du théorème 1 :

Satisfaction de la condition (6). Si $l_S^*(\vec{x}) = 1$ alors $y_0(\vec{x}) \in S$. Or, $d(\vec{x}, y_0(\vec{x})) \leq \beta(h)$ donc $d(\vec{x}, S) \leq \beta(h)$.

Satisfaction de la condition (7). Si $l^*_S(\vec{x}) = -1$, nous avons deux possibilités :
– $y_0(\vec{x}) \in S$
– $y_0(\vec{x}) \notin S$
Si $y_0(\vec{x}) \notin S$, $d(\vec{x}, K_h \backslash S) = d(\vec{x}, y_0(\vec{x})) \leq \beta(h)$, et la condition (7) est satisfaite.

Si $y_0(\vec{x}) \in S$, $\exists y \in y_i(\vec{x})$ tel que $y \notin S$ et $d(\vec{x}, y) \leq \alpha h$. Ainsi, $d(\vec{x}, K_h \backslash S) \leq \alpha h \leq h \leq \beta(h)$.

4 Implémentation et Tests

Dans cette section, nous proposons une méthode d'implémentation qui privilégie la mise en oeuvre par les utilisateurs, au risque d'une perte de précision des résultats. Nous comparons ensuite les résultats obtenus pour deux modèles pour lesquels les Noyaux de Viabilité Théoriques (noté NVT) ont pu être calculés.

4.1 Découplage modèle / algorithme

Dans le jeu de rôle [12] d'aide à la gestion participative SimParc, les joueurs sont amenés à réfléchir sur différents domaines d'une aire protégée, et le jeu comporte deux phases de négociation assez conséquentes. Ceci amène les joueurs à utiliser l'expert viabilité de manière répétée, avec des modifications substantielles de contraintes et éventuellement des paramètres du modèle (voire du modèle lui-même). L'interface graphique utilisateur de l'expert viabilité doit donc être très facile à comprendre et à manipuler. Or le code du modèle est utilisé en profondeur par l'algorithme de Patrick Saint-Pierre [2], pour estimer localement le coefficient de Lipschitz de la dynamique (2), ce qui rend l'utilisation de cet algorithme difficile dans ces conditions. Pour faciliter l'utilisation dans le jeu sérieux, nous proposons un découplage entre l'algorithme de calcul du noyau et l'implémentation du modèle. Le système dynamique est décrit dans un module "implémentation du modèle", et une estimation du coefficient de Lipschitz est fixée pour l'ensemble de l'espace de contraintes : ceci peut entraîner une perte de précision de l'algorithme avec pour résultat une approximation relativement grossière du noyau de viabilité. Dans le module "algorithme" est implémentée la méthode décrite dans les sections précédentes. Pour utiliser l'agent expert et lancer un calcul de noyau de viabilité, les utilisateurs doivent juste définir les paramètres de l'algorithme et sélectionner un modèle.

4.2 Expérimentation

Modèle de croissance de population dans un espace limité. Ce modèle a été élaboré par Maltus et Verhulst. L'état du système est décrit par deux variables, $(x(t), y(t))$: $x(t)$ représente la taille de la population et $y(t)$ son taux d'accroissement. La dynamique à temps discret avec un pas de temps dt est la suivante :

$$\left\{ \begin{array}{lcl} x(t+dt) & = & x(t) + x(t)y(t)dt \\ y(t+dt) & = & y(t) + u(t)dt \; avec \; |u(t)| \leq c \end{array} \right. \quad (10)$$

où c est la variation maximale du taux d'accroissement à chaque pas de temps.

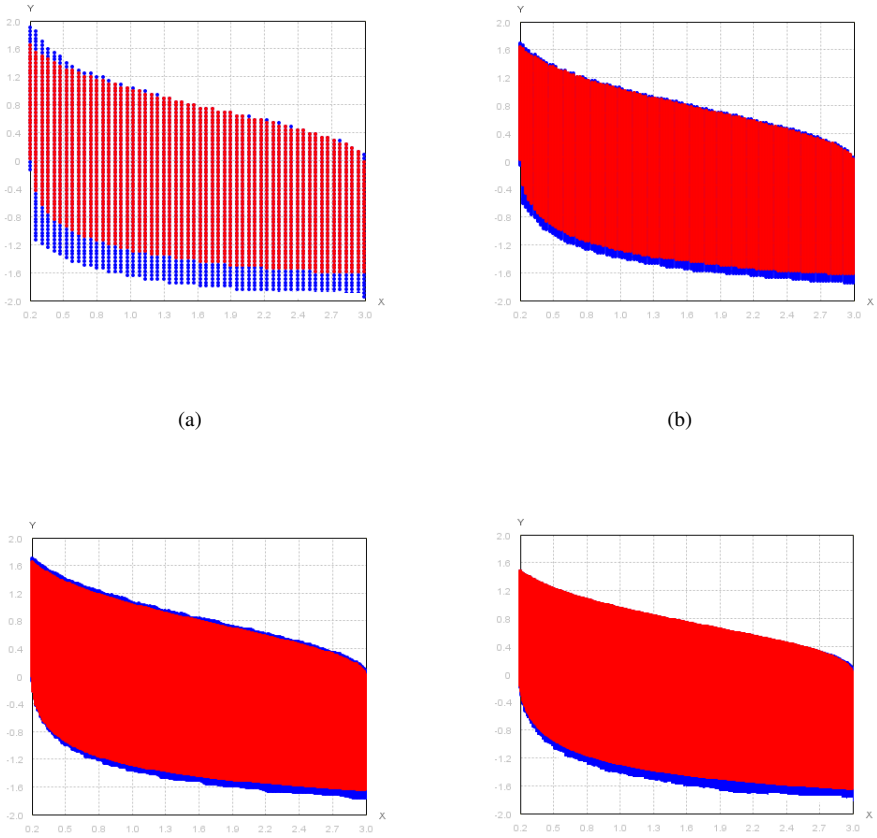
Aubin [14] rajoute des contraintes sur la taille de la population qui doit rester entre deux bornes : l'ensemble des contraintes est ainsi défini par $K = [a, b] \times R$, avec $a > 0$. Ce système a un noyau de viabilité théorique (quand $dt \to 0$), qui est étudié par Aubin [14] :

Viab(K) =
$$\left\{ (x, y) \in R^2 |\; x \in [a; b], y \in [-\sqrt{2.c.log(\tfrac{x}{a})}; \sqrt{2.c.log(\tfrac{b}{x})}] \right\}$$

Nous prenons comme valeurs des paramètres $a = 0$, $b = 3$, $c = 0.5$. Le tableau 1 rassemble les résultats de quatre simulations avec des nombres de points par dimension croissants : tous les noyaux calculés représentent en volume entre 55% et 60% du volume de l'ensemble des contraintes. La fidélité au NVT (mesurée par l'écart en volume entre le NVT et le noyau calculé) est supérieure à 90% et elle augmente avec le nombre des points de la grille. Nous avons également comparé le temps de calcul avec celui du logiciel Kviar [3], le temps de calcul est toujours plus faible et le gain augmente avec le nombre de points. La figure 2 représente les résultats des simulations du tableau 1 avec $dt = 0.1$. Les résultats que nous avons obtenus sont sous forme d'un ensemble de points que nous notons *NoyauApp*. L'approximation est bien sûr meilleure quand le nombre de points augmente (c'est le théorème de convergence).

Simulations	Simu 1	Simu 2	Simu 3	Simu 4
Nb pts dim x	56	112	448	896
Nb pts dim y	80	160	640	1280
Nb pts total	4480	17920	286720	1146880
NoyauApp/K(%)	59.88	57.19	55.34	55.33
Nb Iterations	18	23	34	34
Taux NVT(%)	89.97	93.33	95.16	95.47
Temps	8s	2m09s	22m	1h40m
Temps(Kviar)	19s	2m59s	1h59m	>10h

TAB. 1: Toutes les simulations utilisent les valeurs $dt = 0.1$ et $\alpha = 0.5$. Pour chaque simulation nous acons renseigné le nombre de points selon x, le nombre de points selon y, le nombre de points de la grille (x*y), le volume du noyau de viabilité approché en pourcentage du volume de l'ensemble des contraintes (nombre de points de *NoyauApp*/ nombre de points total), le nombre d'itérations avant l'arrêt de l'algorithme, le volume du NVT en pourcentage du volume du noyau de viabilité approché (nombre de points communs au *NoyauApp* et au NVT) / (nombre de points de *NoyauApp*). Les 2 dernières lignes renseignent les temps de calcul respectifs de notre algorithme et de Kviar. Toutes les simulations sont effectuées avec un processeur Core 2,13 GHz d'Intel (TM) 2 CPU, 2 Go RAM.



(a) (b)

(c) (d)

FIG. 2: Les approximations du noyau de viabilité du problème de population avec $\alpha = 0.5$ et $dt = 0.1$ du tableau 1. L'axe horizontal représente la taille de la population (x) et l'axe vertical son taux d'accroissement (y). L'ensemble des contraintes est le rectangle noir. Le noyau de viabilité théorique (NVT) est représenté en rouge. Le noyau de viabilité calculé approche le NTV par l'extérieur, les erreurs sont en bleu. Les graphiques (a) à (d) correspondent aux simulations 1 à 4 respectivement.

Modèle de consommation. Ce problème est étudié par Aubin [1], le problème est modélisé en deux dimensions $x(t)$ et $y(t)$. $x(t)$ représente la consommation d'une matière première et $y(t)$ représente son prix. L'ensemble de contraintes est $K = [0; b] \times [0; e]$. La variation des prix entre deux pas de temps est déterminée par un variable de contrôle $u(t)$ et bornée par le paramètre c. La dynamique du système représente la consommation d'une matière première, freinée par les prix :

$$\begin{cases} x(t+dt) &= x(t) + (x(t) - y(t))dt \\ y(t+dt) &= y(t) + u(t)dt \; avec \; |u(t)| \leq c \end{cases} \quad (11)$$

Pour ce système simple, il est possible de déterminer analytiquement le noyau de viablité [1] (quand $dt \to 0$) :

Viab(K) =
$\left\{(x, y)|\ x \in [a; b], y \in [x - c + c.exp^{-\frac{x}{c}}; x + c - c.exp^{\frac{x-b}{c}}]\right\}$

Nous prenons $b = 2$, $c = 0.5$ et $e = 3$. La figure 3 représente deux simulations d'approximation du noyau de viabilité avec $dt = 0.1$.

Dans le tableau 2, tous les volumes des noyaux approchés représentent entre 25% et 27% du volume de l'ensemble des contraintes, la fidélité au NVT est supérieure à 97%, et elle augmente avec le nombre des points. Le temps de calcul est beaucoup plus faible que celui de Kviar quand le nombre de points augmente. La figure 3 représente les résultats des simulations du tableau 2 avec $dt = 0.1$.

Simulations	Simu 1	Simu 2	Simu 3	Simu 4
Nb pts dim x	40	160	480	960
Nb pts dim y	60	240	720	1440
Nb pts total	2400	38400	345600	1382400
NoyauApp/K(%)	26.29	25.35	25.31	25.28
Nb Iter	13	33	34	36
Taux NVT(%)	97.01	99.5	99.92	99.94
Temps	9s	4m01s	22m	1h13m
Temps(Kviar)	17s	11m54s	2h21m	> 10h

TAB. 2: Tous les paramètres du tableau sont les mêmes que pour le tableau 1. La méthode de classification proposée ici est plus rapide et ce d'autant plus que le nombre de points augmente.

5 Conclusion

Nous avons présenté un algorithme d'approximation du noyau de viabilité, basé sur une procédure de classification de type "plus proche voisin". Nous avons montré que l'approximation obtenue convergeait vers le noyau de viabilité exact lorsque le pas d'espace tend vers 0. Nous avons comparé les résultats de notre algorithme avec les solutions exactes dans le cas de deux modèles pour lesquelles les noyaux exacts ont pu être calculés. Nous avons montré dans ces deux cas que l'écart au noyau théorique est inférieur à 10 % et décroît avec le nombre de points.
En outre, le temps de calcul, pour les simulations effectuées, est divisé par deux et jusqu'à dix suivant les cas, par rapport au logiciel Kviar qui utilise des SVMs comme fonctions de classification. Cela permettra de rendre le jeu sérieux SimParc plus fluide en diminuant le temps d'attente de l'analyse de viabilité.
Pour l'avenir, nous envisageons d'encapsuler le module de l'algorithme avec une production dynamique de code pour intégrer le calcul de l'estimation locale du coefficient de Lipschitz, qui dépend lui du module modèle. Ceci permettra de rétablir la précision de calcul qui est perdue actuellement. Nous envisageons aussi une interface spécifique pour l'analyse des résultats produits par l'agent expert viabilité. Cette interface permettra de suivre des trajectoires particulières du système dynamique dans le noyau de viabilité, et aussi de construire des trajectoires viables suivant des stratégies de contrôle préétablies. Elle permettra aux joueurs d'argumenter sur les résultats de l'analyse de viabilité.

Références

[1] J.P. Aubin, *Viability theory,* Birkhäuser, 1991.

[2] P. Saint-Pierre, Approximation of the Viability kernel, *Appl. Math. Optim.*, Vol. 29, pp. 187-209, 1994.

[3] G. Deffuant, L. Chapel, and S. Martin Approximating Viability Kernels With Support Vector Machines, *IEEE Transactions on automatic control*, Vol. 52, 5, pp. 933-937, 2007.

[4] P.A. Coquelin, S. Martin, and R. Munos A dynamic programming approach to viability problems, *Dans Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007.

[5] C. Bene, L. Doyen, and D. Gabay A viability analysis for a bio-economic model, *Ecol. Econ*, Vol. 36, pp. 385-396, 2001.

[6] C. Mullon, P. Curry, and L. Shannon Viability model of trophic interaction in marine ecosystems, *Natural Ressource Model*, Vol. 17, pp. 27-58, 2004.

[7] N. Bonneuil, Making ecosystem models viable, *Bull. Math. Biol.*, Vol. 65, pp. 1081-1094, 2003.

[8] D. Michael and S. Chen, Serious Games Games that Educate, *Train and Inform.* , 2006.

[9] J. Warmerdam, M. Knepflt'e, R. Bidarra, G. Bekebrede, and I. Mayer, Sim-port : a multiplayer management game framework, *in 9th International Conference on Computer Games (CGAMES06), Dublin, Ireland*, 2006.

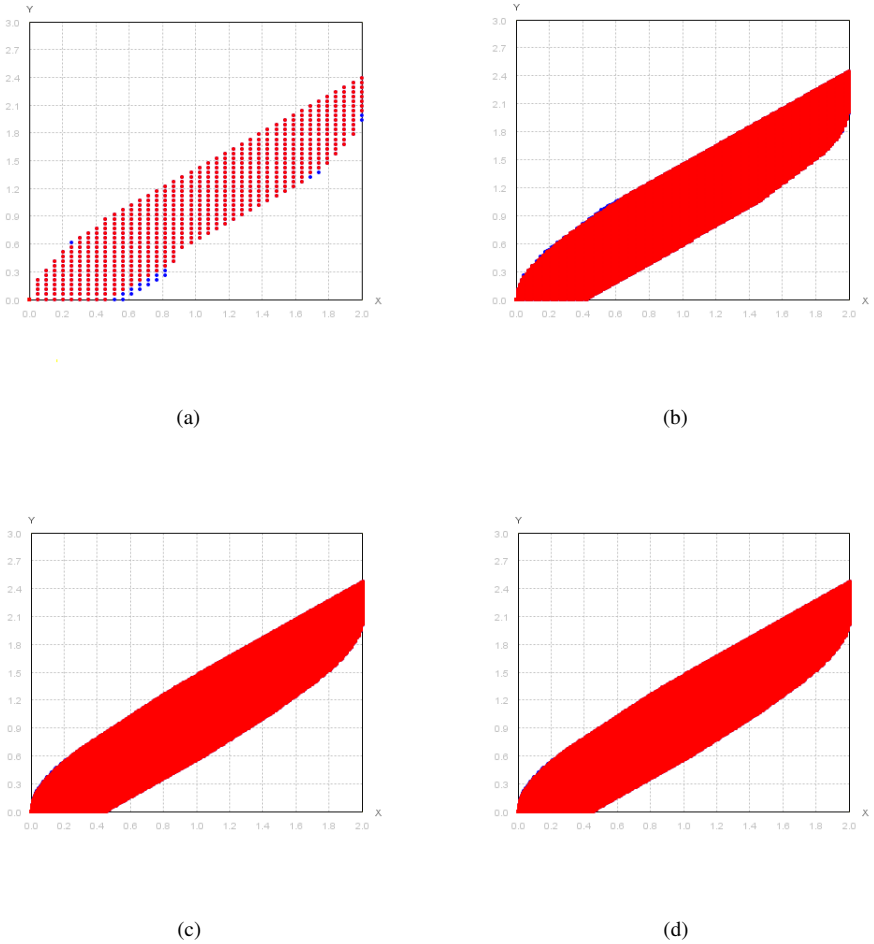
[10] M. De Lara and L. Doyen, *Sustainable Management of Natural Ressources,* Springer, 2011.

[11] L. Chapel, *Maintenir la viabilité ou la résilience d'un système : les machines à vecteurs de support pour rompre la malédiction de la dimensionnalité ?* Thèse, 2007.

[12] Briot, J.P. and Marta, A.I. and Melo, G.M. and Vasconcelos, J.E.F. and Wei, W. and Martin, S. and Alvarez, I., A Serious Game and Artificial Agents to support Intercultural Participatory Management of Protected Areas for Biodiversity Conservation and Social Inclusion., *Culture and Compution 2011*, 2011.

[13] Briot, J.P. and Sordoni, A. and Vasconcelos, J.E.F. and Sebba Patto, V. and Adamatti, D. and Irving, M.A. and Melo, G., ADesign of an Artificial Decision Maker for a Human-based Social Simulation - Experience of the SimParc Project. In David R. C. Hill, Alexandre Muzy, and Bernard P. Zeigler, editors, *Activity-Based Modeling and Simulation*, pages 17-35, Presses Universitaires Blaise-Pascal, 2011.

[14] J.P. Aubin, and P. Saint-Pierre, An introduction to viability theory and management of renwable resources., *Advanced Methods for Decision Making and Risk Management*, pp. 52-96, 2006.



(a) (b)

(c) (d)

FIG. 3: Approximations du noyau de viabilité du problème de consommation avec $\alpha = 0.5$ et $dt = 0.1$ du tableau 2. L'axe horizontal représente la consommation (x) et l'axe vertical le prix (y). L'ensemble des contraintes est le rectangle noir. Le noyau de viabilité théorique (NVT) est figuré en rouge. Le noyau de viabilité s'en écarte pour les points bleus. Les graphiques (a) à (d) correspondent aux simulations 1 à 4 respectivement.