Une architecture orientée comportement, application a la navigation autonome en environnement virtuel

David Hanon

Une architecture orientée comportement, application à la navigation autonome en environnement virtuel

David HANON
L.A.M.I.H. UMR CNRS 8530 – Le Mont Houy, BP 311,
Université de Valenciennes, 59304 Valenciennes cedex, France.
david.hanon@univ-valenciennes.fr

Résumé : Les systèmes multi-agents ainsi que la robotique basée sur les comportements ont montré que la résolution d'un problème peut être répartie entre différentes entités. Chaque entité (encore appelée comportement) est chargée d'un aspect particulier du problème : le respect d'une contrainte ou la réalisation d'un objectif. De nombreux modèles décisionnels distribués ont été développés. Cet engouement s'explique par le fait qu'un système composé d'entités simples et fiables sera plus robuste et plus apte à évoluer dans un environnement complexe.
L'une des difficultés de cette méthode est de dégager une « volonté commune » à partir de préférences ou de contraintes individuelles. Parmi l'ensemble des techniques proposées pour arbitrer les choix des entités, la technique du vote distribué de préférence (VDP) présente de nombreux avantages.
Notre contribution consiste en une modification de la méthode du VDP. Nous proposons de sélectionner les décisions dans un espace continu, d'augmenter le nombre des comportements couramment utilisés et d'utiliser d'autres procédures de vote. Notre application concerne le déplacement de personnages autonomes en environnement virtuel.
Mots Clés : Systèmes multi-agents, Architecture orientée comportement, Réalité virtuelle

1 INTRODUCTION

Les mondes virtuels deviennent de plus en plus réalistes (amélioration des graphismes etc.). Mais, peupler ces mondes virtuels de « créatures intelligentes » reste le défi que de nombreuses équipes de recherche essaient de relever. Le déplacement autonome des acteurs synthétiques est l'un des problèmes majeurs. Sa résolution implique la prise en compte simultanée de deux contraintes. Tout d'abord, lors de ses déplacements, un acteur virtuel doit prendre en considération les actions imprévisibles des autres personnages (utilisateurs ou autres acteurs virtuels). Par ailleurs la réalité virtuelle impose les contraintes du temps réel. Les algorithmes utilisés doivent donc être très efficaces.
Les solutions utilisant la planification produisent des résultats spectaculaires [Pettre, 2002] moyennant un temps de calcul trop élevé pour notre domaine d'application. De plus les changements constants rendent les plans caduques : ils nécessitent une remise à jour continuelle. La planification est donc incompatible avec des environnements complexes et dynamiques. C'est pourquoi nous optons, pour la navigation. Contrairement à la planification, la navigation établit la trajectoire au fur et à mesure du déplacement, à partir des données perçues. L'algorithme présenté dans [Airault, 2004] prend également en compte les positions futures estimées des autres personnages. Dans la pratique, planification et navigation sont utilisées de manière complémentaire, cf. [Lamarche, 2004] par exemple. Nous avons choisi de traiter uniquement la navigation.

Nous avons considéré une vision agent du problème. Chaque acteur synthétique constitue, à l'instar d'un agent, une entité autonome capable de percevoir son environnement et d'agir de manière rationnelle sur celui-ci. La notion d'agent est complexe et dépasse largement le cadre de l'informatique. Marvin Minsky imagine l'esprit humain comme l'assemblage de nombreux agents bien trop simples pour être eux-mêmes considérés comme intelligents [Minsky, 1988]. Il donne la définition suivante : « Agent : Toute partie ou tout processus de l'esprit assez simple à comprendre, même si les interactions entre des groupes de ces agents peuvent produire des phénomènes beaucoup plus difficiles à comprendre » De manière plus consensuelle « l'agent est une entité qui agit ». Comme le montre la figure 1, un agent réalise trois fonctions principales : perception, décision et action.

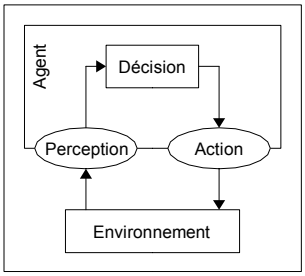

Figure 1 : Cycle minimal d'un agent.

L'architecture d'un agent doit permettre d'intégrer de manière cohérente et efficace les trois fonctions principales. Une architecture modulaire favorise

l'amélioration, l'ajout et la réutilisation des modules. La division en modules s'effectue selon deux approches : fonctionnelle ou comportementale. L'approche fonctionnelle est la plus intuitive ; chaque module est responsable de la réalisation d'une fonction dont le résultat sera utilisé par le module suivant. L'exécution des modules est donc séquentielle, par exemple : percevoir, modéliser l'environnement, décider, et agir. Ce type d'architecture est aujourd'hui relativement désuet. En particulier parce que la modélisation de l'environnement est complexe et que l'approche symbolique a montré ses limites. Les architectures orientées comportement ont une exécution beaucoup plus parallèle ; elles seront détaillées dans la section suivante.

2 ARCHITECTURES ORIENTEES COMPORTEMENT

2.1 Principe général

Le mode de conception orienté comportement trouve ses origines dans des théories sur l'intelligence [Minsky, 1988] et la robotique de la fin des années 80 [Brooks, 1986], [Maes, 1989] et [Arkin, 1989]. Les architectures basées sur les comportements sont divisées en couches le plus souvent réactives. Comme le montre la figure 2, chaque couche correspond à un comportement spécifique. Chaque comportement est activé par un ensemble de stimuli et propose une réponse.

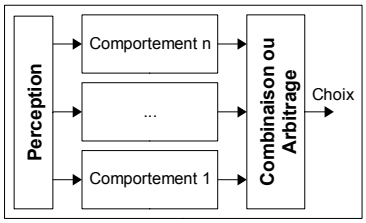

Figure 2 : Architecture orientée comportement.

Le comportement global de l'agent résulte de l'interaction des comportements qui le composent. La distribution en comportements permet de prendre en compte le fait que l'agent :

- réponde à différents stimuli,
- possède différents buts,
- mène plusieurs activités en parallèle.

Le paragraphe suivant liste les différentes stratégies utilisées pour sélectionner une action parmi les propositions concurrentes. Un comportement doit prendre le contrôle de l'agent lorsque :

- l'exécution de ce comportement répond à l'un de ses besoins importants,
- l'objectif de ce comportement est facilement réalisable dans le contexte du moment. Il s'agit alors d'un choix plus opportuniste.

2.2 Mécanismes de coordination d'actions

Le point essentiel des architectures orientées comportement réside donc dans le choix d'un système de coordination capable de sélectionner l'action à accomplir. Le système sélectionnant le ou les comportements pertinents peut être plus ou moins complexe. Les mécanismes de coordination d'actions peuvent être classés en deux grandes familles [Pirjanian, 1999] [Dalgalarrondo, 2001] : l'arbitrage et la fusion de commandes. Les méthodes classées dans le groupe « arbitrage » sélectionnent un comportement pertinent qui va prendre le contrôle de l'agent. La fusion de commandes consiste à choisir une action en prenant en compte les différentes propositions des comportements.

Les mécanismes d'arbitrage sélectionnent l'action proposée par le comportement actif qui a la plus haute priorité (principe utilisé dans la subsomption, par exemple [Brooks, 1986]) ou le comportement qui est reconnu comme le plus pertinent en fonction du contexte. La stratégie nommée "winner-take-all" (le gagnant emporte la mise) [Maes, 1989] suit également ce principe. L'inconvénient de cette méthode réside dans le fait que le choix est celui d'un seul comportement et que les autres sont totalement ignorés.

La fusion de commandes consiste à prendre en compte toutes les recommandations d'actions et à les combiner. Par exemple, si deux comportements effectuent des propositions différentes, alors l'action retenue est une troisième action qui constitue une sorte de compromis entre les deux propositions. Cette stratégie est particulièrement adaptée aux systèmes à base de champs de forces [Zeghal, 1993]. Pour les détracteurs de cette approche, "fusionner" des préférences peut aboutir à un résultat qui ne satisfasse aucune préférence. L'approche dite du vote distribué de préférence (VDP) (*distributed preference voting*) tente d'y remédier ainsi : chaque comportement évalue toutes les possibilités selon un critère particulier. L'action choisie est celle qui a obtenu le plus de voix. Cette stratégie et ses variantes ont été étudiées entre autres par [Rosenblatt, 1995], [Sukthankar, 1997] et [Hostetler et al, 2002].

Cette démarche présente dans son principe fondateur un avantage certain : l'action sélectionnée est un compromis acceptable pour la majorité des comportements. Ce n'est pas le cas ni pour « le gagnant emporte la mise » ni pour la fusion. Le paragraphe suivant apporte plus de détails sur cette stratégie.

2.3 Vote distribué de préférence

Le VDP a été utilisé par Hostetler [Hostetler, 2002] pour simuler des groupes de piétons mais ses origines proviennent de la simulation de trafic routier [Sukthankar, 1997] et initialement de la robotique [Rosenblatt, 1996].

L'action réalisée par l'agent est choisie parmi un ensemble fini de possibilités (« action space » dans la littérature). Sukthankar et Hostetler ont limité cet ensemble à neuf possibilités. Cet espace d'action retenu pour le déplacement d'un agent est représenté sur la table 1. Il correspond aux trois possibilités pour chacun des deux paramètres, vitesse et direction.

	Direction		
Vitesse	plus vite à gauche	plus vite tout droit	plus vite à droite
	identique à gauche	identique tout droit	identique à droite
	moins vite à gauche	moins vite tout droit	moins vite à droite

Table 1 : Espace d'action pour le déplacement.

La sélection d'actions est distribuée entre ces différents comportements de la manière suivante : chaque comportement attribue une note à chaque option. Les notes sont des décimaux de l'intervalle [-1 ; 1]. Par exemple –1 exprime le refus, 0 l'indifférence et 1 l'adhésion totale. Chaque comportement se voit attribuer, expérimentalement ou par apprentissage, un coefficient qui traduit selon les auteurs son importance. Les notes sont multipliées par ces coefficients avant d'être additionnées afin d'obtenir une note globale pour chaque option. L'option dont la note est la plus haute est retenue. Sukthankar introduit la notion de veto (qui n'est pas utilisée par Hostetler). Si un comportement juge une possibilité très néfaste, il dépose un veto contre cette possibilité : elle ne pourra pas être sélectionnée, même si elle recueille la meilleure note. Dans le contexte de la navigation autonome, cette notion évite les sorties de route et les collisions.
La figure 3 montre un exemple de l'utilisation de cette architecture par Hostetler pour contrôler un piéton virtuel. Le comportement d'inertie sur cette figure permet de limiter les tremblements du personnage. En effet ce comportement vote en permanence pour le choix effectué au pas de calcul précédent, par conséquent il s'oppose à toute variation et limite le phénomène d'hésitation. Le problème d'hésitation se pose à chaque méthode de sélection d'action. Il est lié au fait paradoxal qu'un agent doit être capable de persévérer dans ses choix mais aussi de les remettre en cause.

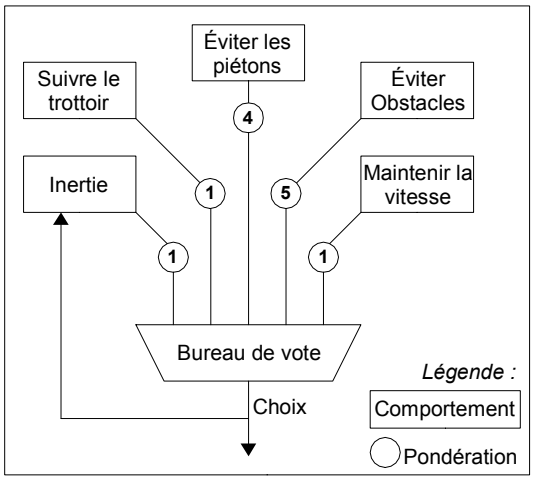

Figure 3 (d'après [Hostetler 2002]) :
Exemple d'application du VDP.

2.4 Synthèse

Le mécanisme d'arbitrage est le point essentiel des architectures orientées comportement. Le vote distribué de préférence présente divers avantages. Il permet à chaque comportement de s'exprimer sur plusieurs options et de choisir une action qui satisfasse un maximum de comportements. Les solutions ne sont pas optimales mais calculées rapidement. L'utilisation de veto garantit que la décision finale ne soit pas extrêmement mauvaise sur un critère. Cela permet d'ajouter des comportements tout en étant certain de ne pas trop dégrader les performances de l'agent. En effet, l'ajout de comportements modifie l'évaluation des options mais les veto priment et assurent une cohérence.
Il faut également noter que ce type d'architecture est utilisable dans d'autres applications. Rosenblatt l'a utilisée pour la navigation un robot sous-marin (problème 3D) [Rosenblatt, 2002] ainsi que pour gérer l'accès concurrent à une ressource.

Quatre critiques peuvent cependant être formulées. Elles concernent l'espace d'action, les pondérations des comportements, leur granularité ainsi que la méthode de vote.

Limiter, comme dans la bibliographie, les possibilités à 9 actions est beaucoup trop restrictif et arbitraire car il y en a souvent beaucoup plus. De plus il faut choisir quelles actions seront pertinentes. Dans le cadre de déplacements, la vitesse et le cap d'un agent étant continus il existe une infinité de choix. Il faut alors choisir un pas d'accélération et de rotation. Si un pas est trop petit, le mobile n'est pas réactif, l'agent ne réagit pas assez vite. Si le pas est trop grand alors les variations de cap et de vitesse seront très fortes. De plus un phénomène permanent d'hésitation risque de se produire puisqu'il est impossible de négocier correctement une trajectoire courbe en tournant avec des angles fixes.

La pondération des comportements soulève aussi quelques problèmes. La figure 3 montre par exemple un rapport de 5 entre les poids de certains comportements. L'écart est si important que les décisions de quelques comportements prédominent largement. Dans ces conditions le choix ne peut être considéré comme un « bon » compromis. Le VDP perd donc son avantage sur les autres méthodes d'arbitrage ou de fusion. Selon les auteurs, la prédominance naturelle de certains comportements suffit à justifier cette différence d'influence. Cet argument est probablement justifiable dans certains cas mais difficilement dans l'application présentée. Selon la figure 3, éviter les piétons est plus important que de suivre le trottoir ou d'éviter un obstacle. Dans ce cas l'évitement d'un autre piéton peut provoquer une sortie de route ou une collision avec un obstacle. Par ailleurs le choix des pondérations peut s'avérer délicat. Hostetler attribue expérimentalement les pondérations. Un algorithme génétique est utilisé dans [Sukthankar, 1997]. Dans ces deux cas, une variation des situations auxquelles sont soumis les agent n'impose t-elle pas de modifier les pondérations ? De même dans le cas de l'ajout d'un nouveau comportement ? On comprend l'intérêt de l'algorithme d'apprentissage. Intérêt cependant limité car l'algorithme génétique fonctionne nécessairement hors ligne.

La granularité des agents réalisés est également discutable. La granularité d'un agent est relative à la quantité et à la complexité des comportements qui le composent. Un agent dont les comportements sont complexes et peu nombreux, possède une granularité élevée. Inversement, un agent composé de comportements simples et nombreux, est dit de granularité faible. La granularité des agents mis en œuvre par Hostetler est souvent très grande ; les problèmes dont leurs comportements ont la charge sont très complexes. Or le VDP s'inscrit dans une volonté de distribution et de réactivité, qui ne peut être compatible avec des granularités élevées (cf définition d'agent par Minsky au paragraphe 2). Pour cela il faut proscrire l'usage de comportement tel qu' « Eviter les piétons » (figure 3). Par ailleurs pour simplifier ce comportement (a priori complexe) l'agent présenté dans [Hostetler, 2002] ne prend en compte que l'obstacle le plus proche.

Nous avons déjà évoqué l'espace d'action, le veto et la pondération, mais le reste de la procédure de vote mérite d'être clarifié. La procédure employée dans la bibliographie est toujours celle du vote par estimation moyenne. Or il existe une multitude de procédures de vote et la méthode par estimation moyenne est très discutable. En particulier il faut que l'attribution des notes soit parfaitement équitable. Le concepteur doit adopter une échelle de notation et la respecter. Ce n'est pas évident si plusieurs personnes interviennent sur le projet. Les pondérations interférent également avec la valeur des notes. Enfin, un comportement « intelligent » devrait avoir tendance à sur-noter les options qui lui semblent intéressantes et sous-noter les autres afin d'augmenter les chances de voir sa préférence retenue. Or ce n'est ni souhaitable, ni simple à réaliser

Le paragraphe suivant présente les modifications que nous avons retenues afin de pallier à ces inconvénients.

3 AMELIORATION DU VPD

3.1 Idées générales

Le premier axe concerne l'espace d'action. Nous avons vu que restreindre les possibilités alors qu'il en existe une infinité n'est pas souhaitable. C'est pourquoi nous utilisons un domaine continu pour l'espace d'action. Si l'on considère l'exemple d'un mobile dirigé en direction et vitesse alors l'espace d'action est un sous ensemble de R^2 : {Cap ; Vitesse } avec le cap choisi dans l'intervalle]-180 ; 180] et la vitesse dans [Vitesse_Min ; Vitesse_Max]. L'espace d'action est continu mais concrètement, les comportements proposent les options « candidates au vote ». Pour garder un des avantages du VDP chaque comportement en propose autant qu'il le souhaite. En effet, plusieurs options permettent d'atteindre un objectif. Par ailleurs, un comportement ne considère qu'une partie du problème et ne peut évaluer ses options par rapport au contexte global.
Un comportement peut être responsable du respect d'une contrainte. Il est simple de déduire quelles options conduiront l'agent à la violation une contrainte. Un veto permet de s'opposer à un ensemble d'options. Afin d'imposer le respect de ses contraintes chaque comportement dépose autant de veto que nécessaire.

La question de la granularité des comportements doit être considérée en même temps que leurs pondérations. Les agents doivent être de granularité plus fine. Nous réalisons, comme le montre la figure 4, une distribution plus fine des tâches. Les comportement sont donc multipliés et simplifiés. Nous n'attachons plus les comportements à un aspect général du problème mais bien à un aspect très particulier de celui-ci. Pour cela, nous choisissons de dupliquer les comportements. Par exemple, l'unique comportement « éviter les piétons » (figure3) est remplacé par autant de comportements « éviter le piéton Pi » (figure 4) que de piétons pris en compte. La conséquence est que le nombre des comportements devient variable, la structure décisionnelle de l'agent s'adapte au contexte. Les pondérations sont supprimées car elle ne sont plus nécessaires.

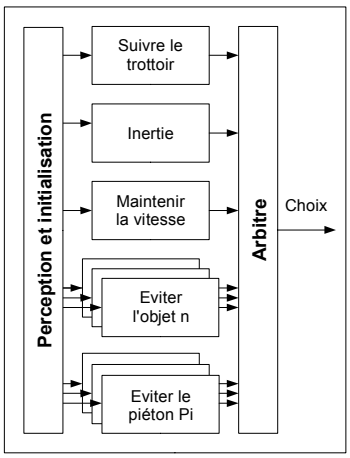

Figure 4 : Granularité « plus fine ».

Pour assurer la simplicité et la reproductibilité des choix nous utilisons les méthodes de vote par classement : chaque comportement classe les solutions selon le critère qui lui est propre. La méthode utilisée pour établir un classement final peut être complexe mais dépend uniquement de l'arbitre. Le fonctionnement de l'arbitre sera discuté par la suite.

3.2 Comportements et données

3.2.1 Comportements

Le modèle décisionnel est distribué entre différents comportements qui gèrent des aspects spécifiques du problème (Cf figure 5). Chaque comportement possède ses propres capacités de perception et décision. Il peut proposer des options afin de réaliser un but ou déposer des veto pour faire respecter une contrainte. Il participe également à la sélection des options en classant les options proposées par l'ensemble des comportements.

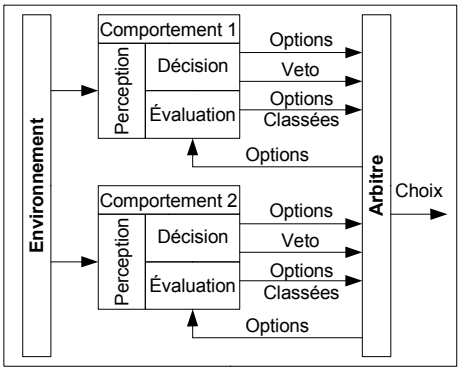

Figure 5 : Distribution entre deux comportements.

Les paragraphes suivants détaillent les données manipulées : les options et veto.

3.2.2 Les options

Comme le montre la formule (1), chaque option o_i est caractérisée par sa valeur vo_i et sa dénomination do_i.

$$o_i = (vo_i ; do_i) \quad (1)$$

Le but de modèle décisionnel est de calculer le vecteur de commande à appliquer à l'agent. La valeur d'une option (formule 2) est donc un vecteur dont chaque composante correspond à une variable de commande et appartient à un domaine D_j.

$$vo_i = (vo_{i,1} ; vo_{i,2} ; \dots ; vo_{i,n}) , \forall\ vo_{i,j} \in D_j \quad (2)$$

Comme le montre la formule (3), les variables de commande de notre application sont la direction et la vitesse de l'agent. Leur domaines de définition sont respectivement]-180 ; 180] et [V_Min ; V_Max].

$$vo_i = (vo_{i,1} ; vo_{i,2})$$
$$vo_{i,1} \in D_1 =]-180 ; 180]$$
$$vo_{i,2} \in D_2 = [V\_Min; V\_Max] \quad (3)$$

3.2.3 Les veto.

La description des veto est proche de celle des options, elle est exprimée par les formules (4) et (5). La valeur des veto est un vecteur d'intervalles. Chaque intervalle appartient au domaine de définition de la variable de commande correspondante.

$$v_k = (vv_k ; do_k) \quad (4)$$
$$vv_k = ([vv_{k,1} ; vv_{k,1}'] ; \dots ; [vv_{k,n} ; vv_{k,n}']) ,$$
$$\forall\ [vv_{k,j} ; vv_{k,j}'] \in D_j \quad (5)$$

Une composante d'un veto peut ne pas avoir d'importance. Dans ce cas elle est notée "*" mais sa valeur « réelle » est égale au domaine de définition de la variable de commande associée. Cette notion n'existe pas dans SAPIENT [Sukthankar, 1997] car le nombre d'actions est très limité (9 cas). Dans ce cas, il est possible de s'opposer à toutes les options d'une même ligne ou d'une même colonne de la table 1.

Dans l'exemple du déplacement (formule 3), un veto peut servir à :

- interdire de se diriger dans un secteur angulaire $v_1 = ([10 ; 20] ; *)$
- limiter la vitesse : $v_2 = (* ; [10 ; V\_Max])$
- limiter la vitesse dans une direction donnée $v_3 = ([10 ; 20] ; [10 ; V\_Max])$

Les dénominations des options et veto sont codées d'une manière quelconque (chaîne de caractère). Nous les utilisons lors la mise au point de nos programmes mais nous souhaitons qu'elles servent de base pour greffer un module symbolique à notre architecture (cf. perspectives).

Nous avons présenté les comportements ainsi que les données manipulées par l'architecture. Nous proposons lors du paragraphe suivant de présenter de quelle manière ceux-ci s'articulent lors de la sélection d'action.

3.3 Algorithme de sélection d'action

La figure 6 présente les grandes lignes du fonctionnement du modèle décisionnel que nous proposons. L'algorithme générique comporte 5 étapes : (1) l'initialisation des comportement, (2) la réception des options et veto, (3) l'application des veto, (4) le vote des comportements et enfin (5) le choix en fonction des votes. Nous détaillons ces étapes dans les sous-sections suivantes.

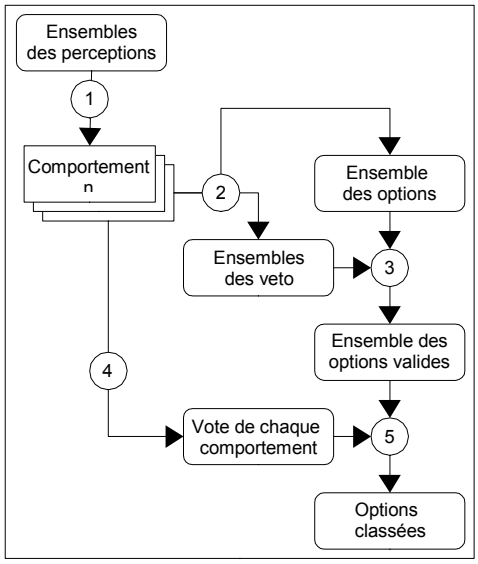

Figure 6 : Algorithme de sélection d'action.

3.3.1 Etape 1, Initialiser les comportements

Les comportements sont instanciés dynamiquement et leur nombre est variable. Certains comportements (comme le comportement d'inertie cf. §2.3) sont toujours utiles. D'autres doivent être instanciés en fonction du contexte. Par exemple, à l'approche d'un obstacle un comportement sera ajouté ; il sera supprimé de l'architecture une fois que l'agent s'en éloigne. De manière générale, le rôle cette étape 1 est d'assigner un comportement pour chaque objet intéressant dans le contexte de la tâche réalisée.

3.3.2 Etape 2, Recueillir les options et veto

Les comportements ont été instanciés, il faut recueillir leurs propositions. La fonction correspondante de chaque comportement doit donc être exécutée, notons simplement que cela peut être fait en parallèle. Il faut également rappeler que différentes techniques peuvent être utilisées à l'intérieur des comportement des plus simples (stimulus réponse) aux plus complexes (Système expert).

3.3.3 Etape 3 , relever les options valides

Cette étape est la première phase de sélection des options qui on été proposées à l'étape précédente : Il s'agit d'éliminer les solutions qui ne respectent pas les veto. Une option n'est pas valide s'il existe au moins un veto pour lequel toutes les composantes de l'option appartiennent aux intervalles correspondants de ce veto. Si l'on note X l'ensemble des options valides le processus de sélection peut se noter de la manière suivante :

$$\forall o_i \, (\exists \, v_k \, (\forall \, j \, ; \, vo_i \in [vv_{k,j} \, ; \, vv_{k,j}']) \Rightarrow o_i \notin X) \quad (6)$$

Remarque : Le corollaire de la formule 6 est le suivant : Si pour chaque veto au moins une composante d'une option n'appartient pas à l'intervalle correspondant dans le vecteur de ce veto alors cette option est valide.

Exemple : Gardons les domaines de définition de la formule 3. Soit les veto de valeur :

- vv_1 = ([10 ; 30] ; *) et
- vv_2 = (* ; [10 ; 20]).

Soit les options :

- vo_1 = (20 ; 5) ; o_1 est éliminée par v_1.
- vo_2 = (20 ; 15) ; o_2 est éliminée par $v_1$et v_2
- vo_3 = (50 ; 15) ; o_2 est éliminée par v_2
- vo_4 = (60 ; 5) ; o_3 reste valide.

Par conséquent X= $\{o_3\}$

3.3.4 Etape 4, évaluation des options

Les options les plus mauvaises ont été éliminées lors de l'étape précédente, les options valides doivent encore être évaluées pour en choisir une à l'étape 5. Comme le système par notation ne nous satisfait pas, nous adoptons un système par classement. Les comportements classent les options valides par rapport un critère. Ce critère correspond à un objectif à atteindre ou une contrainte à respecter

Soit par exemple l'objectif définit par « avancer dans une direction privilégiée θ ». Dans ce cas o_i se classe avant o_j si $vo_{i,1}$ est plus proche de θ que $vo_{i,2}$ En faisant abstraction du modulo ceci se note :

$$|vo_{i,1} - \theta| < |vo_{i,2} - \theta| \Rightarrow o_i > o_j \quad (7)$$

Il en va de même pour des objectifs tels que s'écarter d'une direction privilégiée, se recentrer dans la voie ou s'écarter de l'ordonnée curviligne d'un obstacle.

3.3.5 Etape n°5 : Arbitrage final

Chaque comportement a classé les options. Diverses méthodes peuvent être utilisées pour désigner celle qui est globalement la meilleure : le vote alternatif ou la méthode de Borda sont les plus simples. La méthode de Condorcet est plus longue et n'aboutit pas toujours.

Si on utilise la méthode de Borda, on attribuera pour chaque option 1 point à chaque fois qu'elle est classée en tête par un comportement, 2 points lorsqu'elle est seconde, 3 points pour chaque $3^{ème}$ place etc. L'option qui récolte le moins de points est choisie.

Le problème de l'égalité des options (à la fin de toute la procédure) n'est pas mentionné dans la bibliographie. Le vote par estimation moyenne est vraisemblablement moins sujet aux problèmes d'égalité. Mais ce cas survient dans notre application, cela est lié au fait qu'il y ait beaucoup d'options candidates par rapport au nombre de votants. Mais cela est également dû à l'utilisation de points entiers, par rapport aux notes réelles attribuées par l'estimation moyenne. Le paradoxe de « Fredkin » [Minsky, 1988, chapitre 5.5] nous rappelle que ce problème n'est sans doute pas si important qu'il n'y paraît : « Plus deux solutions paraissent aussi attirantes l'une que l'autre, plus il risque d'être difficile de choisir entre elles, alors que dans ce cas, le choix importe d'autant moins. » En accord avec ce principe, les solutions égales sont départagées par le hasard. Un des effets de l'utilisation du hasard est que les comportements des agents similaires ne sont plus rigoureusement identiques.

4 APPLICATION

Le paragraphe précédent détaillait l'algorithme de notre architecture, nous présentons à présent son application au déplacement autonome de piétons virtuels. Lors des développements et des tests réalisés jusqu'à présent seule la direction de l'agent est choisie par le modèle décisionnel, la vitesse reste constante. Les résultats du modèle de piétons virtuels sont par conséquent discutables, mais ils montrent la pertinence de l'architecture.

Les comportements que nous présentons ici sont très simples. Ceci correspond à un besoin de simplification nécessaire lors d'une première phase de test. Ce choix est également celui d'un partisan de l'approche réactive. Les comportements développés sont :

- le suivi d'une voie,
- l'évitement d'un obstacle fixe,
- l'évitement d'un autre agent,
- le comportement d'inertie.
- Le comportement « aléatoire ».

Le comportement de suivi d'une voie est le premier comportement développé. (Dans le cas de notre architecture la construction incrémentale est réellement possible). L'option qu'il propose est la direction de la tangente à l'axe de la voie (cf. figure 7). Les veto éventuellement déposés sont des angles plats (180°) qui interdisent à l'agent de sortir de sa voie. Ce comportement classe les solutions en fonction de leur distance avec sa propre proposition (formule 10).

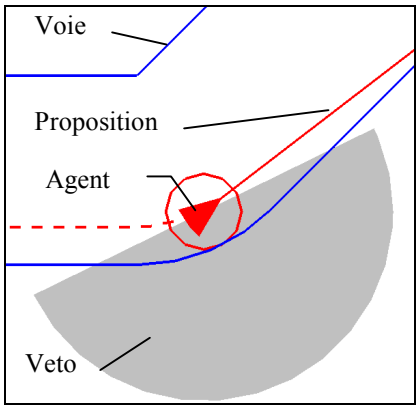

Figure 7 : comportement « avancer sur la route »

La figure 8 montre quelles options sont retenues pour l'évitement d'un obstacle statique : ceux sont les trajectoires tangentes à l'obstacle. Le veto proposé est définit par l'angle entre les deux tangentes. A terme ce veto pourra être plus complexe et prendre en compte les vitesses réduites qui permettent d'approcher l'obstacle sans entrer en collision. Dans notre version actuelle ce comportement n'évalue pas les solutions. Etant donné qu'il invalide toutes celles qui conduisent à une collision, nous n'avons pas jugé cela nécessaire. De plus comme nous souhaitons que ce comportement ne gère que cet aspect particulier du problème, il lui est impossible d'évaluer une solution.

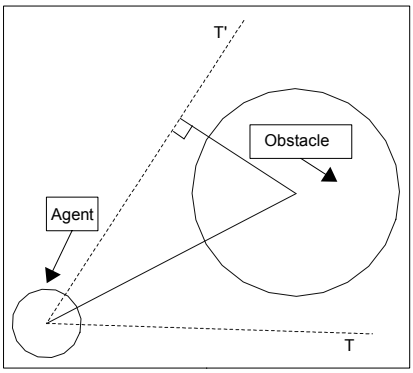

Figure 8 : Evitement d'obstacle statique

Diverses solutions peuvent être retenues pour créer le comportement d'évitement d'un autre agent Il fonctionne ici sur le même principe que l'évitement statique mais il prend en compte la direction de l'agent évité.

Le comportement « aléatoire » peut être utilisé pour montrer la robustesse de l'architecture ou pour sortir l'agent d'une situation de blocage (situation dans pour laquelle aucune option n'est valable, ou que les options sont toutes défavorables).

Les figures 9 et 10 représentent des essais réalisés. Les agents se déplacent sur des voies en s'évitant et en évitant les obstacles statiques. Nous ne pouvons pas encore présenter de résultats numériques. Nous envisageons des comparaisons avec le VPD original et la méthode de fusion d'actions généralisée [Arnaud, 2000].

Différentes situations ont été testées : croisement de flux d'agents, rétrécissement de la route etc. Dans chacun des cas les choix des agents ne sont pas aberrants : ils ne quittent pas la route et il n'y a pas de collision, ce qui n'est pas le cas avec toutes les architectures distribuées. Un avantage certain est que les résultats obtenus n'ont pas nécessité la mise au point expérimentale de pondérations. Cependant les évitements entre piétons conduisent à des trajectoires qui ne sont pas toujours réalistes, c'est un point à améliorer.

Figure 9 : Environnement de tests.

La figure 10 illustre l'apparition d'un blocage entre les agents. Des obstacles fixes obstruent la voie. Les agents qui se croisent ne peuvent passer qu'un par un. Lorsque le nombre d'agents augmentes les blocages ne se résorbent plus. L'application des principes du modèle satisfaction - alstruisme de Simonin [Simonin, 2001] est un axe d'amélioration possible.

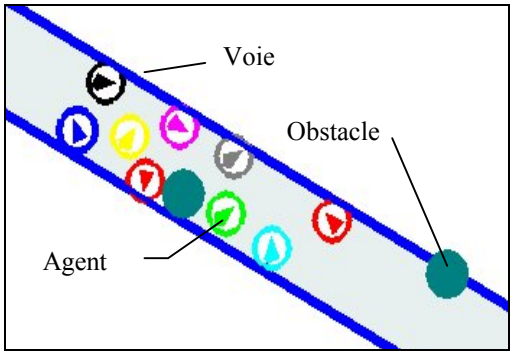

Figure 10 : Blocage.

5 CONCLUSIONS ET PERSPECTIVES

Après avoir expliqué le contexte de notre travail et introduit les architectures orientées comportement nous rappelons les principes du vote de préférence distribué. Nous présentons ensuite l'amélioration que nous en proposons. Nous avons implémenté cette architecture et elle a été appliquée aux déplacements autonomes Elle peut être utilisée pour d'autres applications. Ceci constitue une première perspective. L'ajout d'un module symbolique à notre agent en est une seconde. En effet la dénomination des options constitue un lien possible entre les données numériques manipulées ici et les symboles nécessaires pour contrôler l'agent à un niveau plus abstrait [Grislin, 2005]. L'utilisation conjointe de méthodes réactives et cognitives étant depuis longtemps un axe de progrès potentiel pour les agents.
A un niveau plus appliqué, il reste à effectuer les comparaisons entre des résultats obtenus avec l'architecture originale et notre modification. La modification et l'ajout de comportements est envisageable. Pour résoudre les problèmes de blocages nous pensons qu'il est possible d'appliquer les principes du modèle satisfaction - alstruisme de Simonin [Simonin, 2001].

Bibliographie.

[Airault, 2004] Airault V., Espié S., Lattaud C., Auberlet J-M, "Interaction between pedestrians and their environment when road crossing : a behavioural approach", 24th Urban Data Management Symposium (UDMS'2004), Chioggia, Italy, 27-29 octobre 2004.

[Arkin, 1989] Arkin R. C., "Motor Schema-Based Mobile Robot Navigation", *The International Journal of Robotics Researc*h, vol. 8, n°4, 1989, p. 92-112.

[Arnaud, 2000] Arnaud P., "Des Moutons et des robots ", Presses polytechniques et universitaires romandes, Lausanne, 2000.

[Brooks, 1986] Brooks R. A., "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, RA-2/1, pp. 14-23, 1986.

[Dalgalarrondo, 2001] Dalgalarrondo A., "Intégration de la fonction perception dans une architecture de contrôle de robot mobile autonome", Thèse de l'université de Paris-Sud, Centre d'Orsay, janvier 2001.

[Maes, 1989] Maes P., "The dynamics of action selection", Proceedings of the International Joint conference on Artificial Intelligence, IJCAI-89,1989.

[Hostetler, 2002] Hostetler T., Karrney J., "Strolling Down the Avenue with a Few Close Friends", Eurographics Ireland 2002 Workshop Proceedings, Dublin Ireland, pages 7-14, Mars 2002.

[Grislin, 2005] Grislin-Le Strugeon E., Hanon D., et Mandiau R., "Behavioral Self-control of Agent-Based Virtual Pedestrians", dans F.F. Ramos et als. (Eds.): ISSADS 2005, LNCS n° 3563, pp. 529-537, Springer-Verlag, 2005.

[Lamarche, 2004] Lamarche F. et Donikian S. "Crowds of Virtual Humans : a New Approach for Real Time Navigation in Complex and Structured Environments.", Computer Graphics Forum, 23(3), Eurographics'04, 2004.

[Minsky, 1988] Minsky M., "La société de l'esprit", InterEdition, Paris, 1988.

[Pettre, 2002] Pettre J., "Planification de marche pour acteur virtuel", Rapport LAAS (Laboratoire d'analyse et Architecture des Systèmes) n°02179, mai 2002.

[Pirjanian, 1999] Pirjanian P., "Behavior Coordination Mechanisms – State-of-the-art", Research Report Robotics Research Laboratory, University of Southern California, October 1999.

[Rosenblatt, 1996] Rosenblatt, Rosenblatt J.K., "DAMN: A Distributed Architecture for Mobile Navigation" Thèse de l'université de Carnegie Mellon University , Pittsburgh, Soutenue le 30 septembre 1996.

[Rosenblatt, 2002] Rosenblatt J., Williams, S., Durrant-Whyte, H., "A Behavior-Based Architecture for Autonomous Underwater Exploration". *International Journal of Information Sciences,* vol 145, no 1-2, Sep 2002 pp 69-87.

[Simonin, 2001] Simonin O., "Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique", thèse de l' Université de Montpellier 2, Soutenue le 20 Décembre 2001.

[Sukthankar, 1997] Sukthankar R., "Situation Awareness for Tactical Driving", Thesis of the Robotics Institute, Carnegie Mellon University, Pittsburgh, January 27, 1997.

[Zeghal, 1993] Zeghal K., "champs de Forces Symétriques : un Modèle de Coordination d'Actions Réactive Appliqué au trafic Aérien", Rapport LAFORIA n° 93/14, mai 1993.