La complexité dans les simulations multi-agents

Y. Kubera kubera@lifl.fr
P. Mathieu mathieu@lifl.fr
S. Picault picault@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq Cédex – FRANCE[0]

Résumé

La complexité algorithmique d'une simulation multi-agent résulte de plusieurs sources : complexité intrinsèque du phénomène abordé, compétences cognitives des agents, mais aussi choix de modélisation décidés lors de la conception du modèle distribué et du moteur de simulation sous-jacent (modèles de comportements, d'environnement, de temps, de perception, etc.), chacun étant en outre susceptible d'être implémenté de multiples façons. Il est nécessaire de mesurer les conséquences de chacun de ces choix, de l'analyse à l'implémentation, aussi bien pour optimiser le déroulement de la simulation informatique qui en résulte, pour éviter d'y introduire des biais parfois critiques, et pour obtenir la simulation la plus simple possible d'un phénomène donné.

Mots-clés : *Génie logiciel, Programmation de SMA, Simulation multi-agent*

Abstract

The algorithmic complexity of a multi-agent simulation comes from several sources : the intrisic complexity of the phenomenon to tackle, cognitive skills of agents as well as the modeling choices made while designing the distributed model and the underlying simulation engine (behaviour-, environment-, time-, perception models, etc.), each one being likely to be implemented by multiple ways. It is necessary to explain each of those choices, from analysis to implementation, to optimize the progression of the resulting computer simulation, to avoid the introduction of sometimes critical biases and to obtain a simulation as simple as possible for a given phenomenon.

Keywords: *Software engineering, MAS programming, Multi-agent Simulation*

1 Introduction

Le domaine de la simulation multi-agent regorge de plateformes fort variées, chacune avec ses spécificités, parmi lesquelles l'expérimentateur a du mal à choisir. Néanmoins, il existe de nombreux critères discriminants pour ces plateformes, comme la gestion de l'espace, la gestion des événements, le modèle de communication ou le choix du « tout agent ».

Ces différents critères jouent sur la complexité des modèles simulés par ces plateformes et sur le temps de calcul. La complexité est un terme vague pour lequel un grand nombre de définition existent. Dans la *complexité de modélisation* figurent entre autres la complexité sémantique [4], le choix du niveau de détails du modèle [8] ou la complexité du passage du modèle à l'implémentation [11]. La complexité peut aussi apparaître à la fin du processus de simulation, pour *vérifier et valider* le modèle utilisé. Elle est de même présente lors de *l'interprétation des résultats* : *syntactic complexity*, *observer complexity* dans [4] ou *relative being* dans [7], ...[1]

L'un des intérêts de l'approche agent est de coder des comportements individuels destinés à produire un phénomène collectif. Celui-ci étant atteint, il faut tenter de le produire au moyen du modèle le moins complexe possible, afin d'en faire apparaître les propriétés fondamentales. Étant donné un phénomène à simuler d'un certain niveau de complexité, il est judicieux d'adapter la complexité du modèle et du code *a minima* par rapport aux attentes, pour éviter des développements et des temps de calcul disproportionnés. De plus on peut toujours simuler un modèle au moyen de n'importe quelle plateforme, mais ceci se paie toujours par l'introduction de biais parfois importants. Il est donc crucial, tant lors de la conception du modèle que lors de de son implémentation, d'être capable de mesurer le degré de complexité qui peut être atteint sans biais dans la simulation.

Tout ces points ne peuvent être assurés qu'avec une bonne connaissance de la complexité de la simulation allant du modèle à la plateforme.

[0]Ce travail est co-financé par le FEDER et le Contrat-Plan Etat Région TAC du Nord-Pas de Calais.

[1]Cette liste n'est pas exhaustive, et ces points sur la complexité ne seront pas développés dans l'article

Dans cet article, nous nous intéressons plus particulièrement à la relation entre les choix de conception du simulateur informatique, la complexité du simulateur par rapport aux autres approches possibles, et les biais potentiellement introduits. Pour cela, nous mettons en évidence les parties principales d'un moteur de simulation, puis nous discutons des simplifications du moteur à l'étape de modélisation, et enfin d'optimisations possibles lors de son implémentation, aussi bien en termes de gains en complexité qu'en termes de biais introduits. La partie 2 montre d'abord quelle est la part de complexité inhérente à toute opération de simulation ; la partie 3 fait apparaître la complexité issue des choix de conception des modèles d'agents et comportements ; enfin la partie 4 montre comment l'implémentation de ces modèles peut elle-même introduire des biais, soit en réduisant la complexité atteignable, soit en augmentant indûment le temps de calcul.

2 Formulation générale de la complexité

Il est possible de caractériser la complexité d'une simulation informatique en quantifiant le nombre d'opérations effectuées entre le début et la fin de la simulation (complexité algorithmique). Les origines de cette complexité se situent dans différents aspects du processus de simulation :

- Dans le phénomène étudié.
- Dans la complexité de la cognition des agents (agents tropiques ou hystériques [6], apprentissage, ...).
- Dans la façon de concevoir le modèle du phénomène simulé et du moteur de simulation.
- Dans la façon de réaliser ces deux modèles.

Le phénomène étudié dans la simulation détermine complètement le premier point et contraint fortement le second. Le modélisateur [11] ne peut alors contrôler que les deux derniers points, pour lesquels il est possible de sélectionner les paradigmes les plus adéquats au phénomène étudié et à la cognition des agents, et de faire des choix d'optimisation du simulateur informatique sous-jacent.

Nous utilisons dans le reste de l'article les définitions de *comportement* et de *comportement réalisable* suivantes :

Définition : *Comportement*
Un **comportement** est un ensemble d'actions initiées par un agent appelé **source** s'appliquant à un ou plusieurs agents (appelés **cibles**), déclenché par des **perceptions spécifiques** et soumis à certaines **conditions d'exécution**. Tout agent dispose d'un ensemble de comportements définissant ses possibilités d'actions sur l'environnement et avec d'autres agents.

Définition : *Comportement réalisable*
Un comportement d'un agent x est dit **réalisable** sur un ensemble de cibles T si la condition d'exécution du comportement appliquée à T est vérifiée.

2.1 Moteur et comportements

Dans la majeure partie des simulations informatiques, le moteur de simulation est restreint à l'ordonnancement des agents. C'est en fait sur l'agent que reposent à la fois la définition des comportements possibles et leur sélection et exécution, via une architecture plus ou moins élaborée, avec en général une forte intrication entre l'algorithme de choix et les objets de ce choix. Ainsi, il est difficile d'ajouter un comportement à l'agent, de modifier un comportement, ou de changer la façon dont il est choisi, sans avoir à modifier l'ensemble de l'agent [10].

Les systèmes experts face au même genre de problème ont proposé d'identifier séparément la base de faits (connaissances), la base de règles (règles d'inférence) et le moteur d'inférence. Il existe alors diverses manières de représenter les règles (logique propositionnelle, logique des prédicats, ...), la base de faits (modélisation de l'environnement, de l'état de l'agent, ...) ou le moteur d'inférence (chaînage avant, arrière, mixte, ...), mais une fois ceux-ci fixés, l'ajout de faits ou de règles ne modifie aucune des deux autres composantes du système expert.

De la même façon, il est possible de diviser la simulation informatique en trois entités :

- Le **moteur de simulation** gérant les accès à l'environnement, l'évolution de celui-ci, et l'exécution des actions des divers agents.
- La **sélection de comportements** utilisée par les agents pour connaître le comportement qu'ils doivent adopter en fonction de ce qu'ils perçoivent.
- Les **comportements** correspondant à l'ensemble des actions effectuées par un agent résultant de la sélection de comportements.

Un agent est alors un agrégat de comportements et d'un état interne. L'architecture de sélection de comportements peut alors soit être propre à l'agent (par exemple pour des simulations faisant intervenir des agents hétérogènes), soit être centralisé dans le moteur de simulation si tous les agents utilisent le même type de sélection d'action.

En outre, il devient possible d'étudier séparément la complexité de ces 3 composantes. Dans les paragraphes qui suivent, nous présentons à titre d'exemple un moteur de simulation (figure 1), et la complexité sous-jacente, pour tout type modèle de simulation séquentielle (cf § 3.1 p 5) à agents situés. Même si cette partie est spécifique au temps séquentiel, les problèmes soulevés sont d'ordre plus général et pourront être considérés dans la suite de l'article.

À chaque unité de temps :

1. Mettre à jour l'état de l'environnement
2. Tant que la liste des agents de l'environnement n'a pas été intégralement parcourue :
   (a) Sélectionner un agent x dans la liste n'ayant pas déjà été selectionné
   (b) Percevoir les caractéristiques de l'environnement dans le champ de perception de l'agent x
   (c) Percevoir les agents voisins $V(x)$ présents dans le champ de perception de l'agent x
   (d) Sélectionner un comportement C et son ensemble de cibles T
   (e) Effectuer les actions de C avec T comme cible si une cible est sélectionnée

FIG. 1 – Algorithme d'un moteur de simulation séquentiel

2.2 Le moteur de simulation

A chaque étape proposée dans l'algorithme 1, il est possible de caractériser la complexité par certains critères. Ceux-ci sont déterminés lors de la modélisation et de l'implémentation par les divers choix pouvant être effectués (voir suite de l'article).

Le critère de complexité le plus évident est sans nul doute le temps que dure la simulation (i.e. le nombre de pas de temps dans une simulation à temps discret). En effet, le nombre d'actions effectuées dans la simulation, et donc le nombre d'occurrences de la sélection de comportements, est proportionnel au temps passé dans cette première (figure 1, première ligne).

La mise à jour de l'environnement (figure 1, 1.) est une étape permettant d'effectuer des modifications sur l'environnement, correspondant à son évolution spontanée (i.e. qui n'est pas issue du comportement d'un agent). Sa complexité est fortement dépendante du modèle utilisé et du phénomène modélisé.

Notation : On pose $Cmaj_{env}(t)$ la complexité algorithmique de la mise à jour de l'environnement au temps t.

Notation : On pose $agents(t)$ l'ensemble des agents présents dans la simulation au temps t, et $n_{agents}(t)$ sa cardinalité.

Le nombre d'agents présents dans la simulation a aussi un impact sur la complexité du modèle, et ce pour deux raisons :

- Il est nécessaire de faire agir chaque agent, et donc de parcourir la liste des $n_{agents}(t)$ agents à intervalles réguliers (figure 1, 2.).
- Tout agent devant agir à un même temps t doit avoir la même probabilité d'être sélectionné (équité dans la sélection des agents), afin de ne pas favoriser certains agents par rapport aux autres (figure 1, 2.(a)).

Notation : On pose $Cchoix_{agent}(i, n)$ la complexité de la sélection équitable d'un agent $x(i, t)$ parmi les n-i^e restants. Pour ne pas surcharger la notation, on utilisera par la suite x à la place de l'expression $x(i, t)$.

Pour qu'il puisse agir différemment selon sa situation, l'agent va devoir percevoir son environnement. Cette étape est divisée en deux parties. La première (figure 1, 2.(b)) pose le problème de la perception des caractéristiques et de la représentation interne de l'environnement.

Notation : On pose $Cper_{env}(x, t)$ la complexité de perception des caractéristiques de l'environnement pour un agent x au temps t.

La seconde est la perception des voisins, pour que l'agent puisse sélectionner le comportement qu'il utilisera, et plus précisément les agents cible de celui-ci (figure 1, 2.(c)).

Notation : On pose $Cper_{voisins}(x, t)$ la complexité de perception des voisins de x au temps t, et $n_{voisins}(x, t)$ le nombre de voisins trouvés.

Ensuite, avec la connaissance qu'il a de son environnement, l'agent doit déterminer l'action qu'il va entreprendre au travers d'un mécanisme de sélection de comportement. Cet aspect du moteur de simulation sera discuté dans la partie suivante (figure 1, 2.(d)).

Notation : On pose $Csel(x, t)$ la complexité de la sélection d'un comportement $c(x, T, t)$ pour l'agent x et de ses cibles T au temps t. Pour ne pas surchager la notation, on utilisera par la suite c à la place de l'expression $c(x, T, t)$.

Enfin, l'exécution de l'action va induire une complexité liée à la fois au phénomène simulé, et au type de modèle utilisé (figure 1, 2.(e)).

Notation : On pose $Cact(c, t)$ la complexité des actions du comportement c de x sélectionné au temps t sur les cibles T.

La formule générale de complexité du moteur de simulation C est alors :

$$C = \sum_t \Big(Cmaj_{env}(t) + \sum_{i=1}^{n_{agent}(t)} \Big(Cchoix_{agent}(i, n_{agent}(t)) + Csel(x,t) + Cact(c,t) \Big) \Big)$$

2.3 La sélection de comportement

Une fois connus l'agent source x et son environnement proche, le moteur doit déterminer quelle action effectuer, donnant ainsi naissance à un ensemble d'opérations complexes : la sélection de comportement. Dans le cas général, on ne connaît pas son algorithme, mais il est possible de mettre en avant certaines de ses caractéristiques.

La sélection du comportement s'effectue sur l'ensemble des comportement possibles, mais seuls ceux réalisables sont réellement pris en compte lors de la décision du comportement à adopter. Il est donc possible de séparer la sélection en deux parties distinctes : l'une récupérant l'ensemble des comportement réalisables, l'autre déterminant le comportement à sélectionner parmi ceux-ci.

Définition : *Cardinalité d'un comportement*
La **cardinalité d'un comportement** correspond au nombre de cibles t que ce comportement nécessite.

Pour déterminer si un comportement est réalisable, il est nécessaire de déterminer sur quels agents tester la condition d'exécution. Le principe d'équité dans le choix des cibles imposé par la simulation (pour éviter de favoriser certaines sources) nécessite d'évaluer pour chaque comportement chaque ensemble de cibles T possible. Ainsi, plus la cardinalité est élevée, plus il y a de comportements potentiellement réalisables à tester et donc plus la complexité de cet algorithme est élevée. Le nombre de tests à effectuer est donc fonction du nombre de comportements de l'agent x, ainsi que de la cardinalité de chaque comportement $card(c)$ et du nombre de voisins $n_{voisins}(x,t)$ de l'agent x.

Notation : On pose $Ccond_c(x,T)$ la complexité de la condition du comportement c de x testé sur les cibles T et $n_{realisable}(x,t)$ le nombre de comportements réalisables par x au temps t.

La seconde étape consiste à sélectionner parmi les comportement réalisables celui qui sera effectué par l'agent. C'est ce choix qui rend l'expression de la complexité de la sélection imprévisible, car il n'est à priori pas possible de savoir pour deux comportements réalisables différents lequel choisir.

Notation : On pose $Creal(x,t)$ la complexité de la sélection d'un comportement réalisable parmi les $n_{realisable}(x,t)$ disponibles.

La complexité de la sélection de comportements s'exprime alors sous la forme :

$$Csel(x,t) = \sum_{i=1}^{c} \Big(\sum_T \Big(Ccond_{c_i}(x,T) \Big) \Big) + Creal(x,t)$$

2.4 Conclusion

Cet exemple fait apparaître clairement la séparation entre ce qui est propre au phénomène simulé (cardinalité des comportements, actions effectuées par un comportement, nombre d'agents, nombre de comportements par agent...) et dont la complexité ne peut être réduite (puisque c'est ce qui caractérise la simulation voulue), et ce qui est propre au moteur de simulation, et qu'il est possible de redéfinir selon les besoins.

Dans les parties qui suivent, nous montrons que la complexité ne trouve pas seulement son origine dans ce qui est propre au phénomène simulé, mais aussi dans la finesse de modélisation et d'implémentation de celui-ci.

3 La complexité liée aux modèles

Dans un premier temps, nous allons présenter, parmi les principaux critères de modélisation, ceux dont la finesse de représentation (i.e. proximité du phénomène réel) a une influence significative sur la complexité de la simulation résultante.

Du fait de la multitude des critères modifiant la complexité du simulateur, nous avons fait le choix de ne pas exprimer formellement cette complexité dans la suite de l'article pour ne pas le surcharger.

3.1 Le temps

Les paradigmes temporels dans les simulations offrent des moyens plus ou moins raffinés pour exprimer le comportement des agents, l'évolution des états de ceux-ci, et l'évolution de l'environnement. Il en existe trois principaux.

L'approche Asynchrone. L'approche asynchrone considère le temps comme continu. L'état des agents et de l'environnement évoluent continuellement, et tous les états transitoires de chaque comportement sont visibles [11] : le changement des états n'est pas uniquement déterminé par leur valeur finale, mais par la façon dont ils y parviennent.

L'approche asynchrone la plus commune est l'approche par événements, pour laquelle chaque agent planifie indépendamment l'occurrence de ses actions, et notifie le moteur de simulation de celles-ci par un événement. Le moteur se contente alors d'exécuter les actions lui étant notifiées par un événement. Cette approche réaliste, qu'on peut trouver par exemple pour des robots mobiles, pose le problème des conflits d'évolution continue : si deux événements induisent des évolutions contradictoires (collision des fonctions de transition, [12]), les composer (voir figure 2) ?

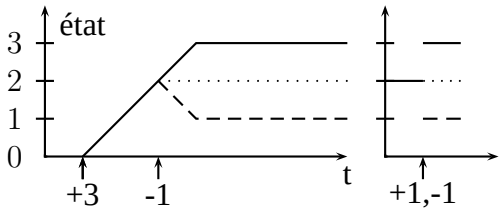

FIG. 2 – Exemple de conflit pour un état de type entier, en temps continu (gauche) et en temps discret (droite). A partir du second événement (représenté par "-1"), la ligne pleine correspond à une évolution qui ignore le second événement, les tirets à une évolution qui se substitue à celle induite par le premier événement, et la ligne en pointillé à une synthèse des deux événements.

La résolution de ce conflit nécessite un algorithme de décision, qui en général n'est ni trivial, ni exempt de biais. Sa complexité est proportionnelle au nombre de conflits au même instant sur l'évolution d'un état, et donc au nombre d'agents et d'actions accédant à un même état.

De plus, il est nécessaire de maintenir la validité des états de tous les agents et de l'environnement à intervalles réguliers, ce qui induit une complexité de mise à jour proportionnelle à la fréquence de celles-ci (élevée dans le cas du temps continu), au nombre d'agents, à leur nombre d'états et au nombre d'états de l'environnement.

L'approche Synchrone. Pour réduire la complexité de l'approche réaliste, il est possible de diminuer la fréquence des mises à jour en approximant le temps de manière discrète. C'est par exemple le cas des automates cellulaires. Cette approche constitue une simplification de la réalité réduisant considérablement le nombre d'opérations de mise à jour, mais ajoute des biais dus à la perte des états transitoires (exemple : oscillations [11]), et la nécessité d'une copie locale de l'environnement, qui n'est pas envisageable dans le cas des simulations large échelle.

L'approche Séquentielle. La résolution de conflits nécessite beaucoup de ressources supplémentaires, aussi bien en mémoire qu'en décisions (résolution de conflits). L'approche séquentielle vise à simplifier la résolution de ces conflits en utilisant le minimum de ces ressources.

Pour cette approche, a chaque pas de temps le comportement des agents est évalué séquentiellement. C'est alors l'ordre d'évaluation qui résoud les conflits. La simplicité de cette approche est au prix de biais non-négligeables, à l'image de la perception de l'environnement à un même pas de temps pouvant différer d'un agent à l'autre, ou la résolution naïve de conflits.

Temps et comportement. Il est possible de donner aux comportements une certaine épaisseur temporelle, et ce indépendamment de l'approche temporelle utilisée. Si cette épaisseur est évidente dans le cas de simulateurs à temps continu, elle l'est moins dans le cas du temps discret, où les actions sont le plus souvent à effet instantané. La gestion de cette épaisseur temporelle complexifie la simulation indépendamment de l'implémentation sous-jacente, puisque le temps nécessaire pour simuler le même phénomène augmente proportionnellement à l'épaisseur temporelle des actions, et qu'il est nécessaire d'effectuer des tests de cohérence supplémentaires (un agent qui agit peut-il être la cible d'un comportement ?, A quel moment peut-on considérer la cible comme ne participant plus à l'action ?, Que doit-on faire si une des cibles disparaît ?, Doit-on vérifier à chaque pas de temps la réalisabilité du comportement ?, ...).

Temps et décision. Dans la réalité, il y a toujours un temps de latence entre la perception et l'exécution du comportement, que l'on peut considérer comme temps nécessaire à la perception d'un phénomène et de la décision du comportement à utiliser en réponse. C'est le cas par exemple des simulateurs de conduite, où le temps de réaction des conducteurs joue un rôle primordial dans leur comportement sur la route.

Dans la plupart des simulateurs, ces deux étapes sont effectuées au même moment pour simplifier les mécanismes mis en œuvre, la séparation n'étant nécessaire que dans des simulations où le temps joue un rôle central (temps continu), ce qui la rend peu utilisée dans le cas du temps discret.

3.2 L'environnement

La modélisation d'un phénomène passe obligatoirement par la modélisation de l'environnement, en tant qu'espace dans lequel évoluent les agents, ou en tant que ressource partagée par l'ensemble des agents. Ces informations peuvent être représentées sous diverses formes.

La gestion de l'espace. La première partie de la modélisation de l'environnement consiste à faire un choix entre environnement discret et environnement continu. Ce choix est nécessaire à l'étape de modélisation, car il détermine en partie comment exprimer les déplacements dans celui-ci (l'autre partie déterminante étant la représentation du temps).

Un environnement continu est proche de la réalité, mais introduit une complexité proportionnelle au nombre d'agents dans la phase de perception à l'implémentation, puisque pour connaître ses voisins, un agent doit mesurer la distance le séparant de chaque autre agent. Son principal intérêt réside dans le fait qu'il permet d'exprimer pleinement la notion de continuité de l'état d'un agent, et donc est adéquat pour une approche temporelle continue. Les simulateurs de trafic ou tout système à lois physiques en ont besoin.

Un environnement discret simplifie le positionnement et les déplacements des agents, en divisant l'environnement en cases contiguës, dont la forme et le nombre de côtés sont variables. La position d'un agent y est représentée par la case sur laquelle il se situe. Cette approche ne permet pas d'exprimer une position transitoire entre deux cases mitoyennes, et introduit donc un biais. Néanmoins elle est souvent suffisante pour les simulations en sciences sociales.

Les ressources de l'environnement. Les ressources de l'environnement peuvent être représentées sous la forme d'agents, permettant ainsi l'unification des entités présentes sur l'environnement, ou au contraire sous la forme d'artefacts spécifiques à la représentation de ces ressources, à l'image de Netlogo[2] qui utilise des patches en plus des « tortues ».

On pourrait croire naïvement que la distinction entre agents et artefacts permet de gagner en temps de calcul, dans la mesure où les artefacts ne disposent d'aucun comportement, d'aucune perception et n'ont donc pas à être pris en compte par la boucle principale du simulateur. Mais une approche « tout agent » correctement implémentée doit justement tenir compte de ces agents dépourvus de comportement et les traiter sans pénaliser la simulation.

3.3 La perception

Pour agir, les agents doivent percevoir leur environnement et les agents proches, ce qui nécessite la définition du modèle de perception utilisé. Il arrive qu'un agent soit omniscient, mais dans la majorité des simulations, le voisinage est plutôt local et correspond à un sous-ensemble de l'environnement.

Ce voisinage peut lui-même être calculé de différentes façons selon que l'agent est orienté ou non. Une simulation de trafic nécessite des agents orientés avec un champ visuel qui s'étend devant eux. C'est typiquement ce que l'on peut obtenir avec NetLogo. Par opposition, le modèle de ségrégation de Schelling [13] ne fait appel qu'à un voisinage de Moore ou de Von Neumann.

On peut en outre imaginer qu'un agent dispose de plusieurs canaux de perception (l'équivalent de la vue ou de l'ouïe), chacun doté de caractéristiques spatiales spécifiques (par exemple le son non orienté mais la vue orientée). La prise en compte de ces critères introduit un niveau de complexité indubitable.

3.4 Les comportements

Les actions exécutées par un comportement constituent un point important de la simulation, puisque ce sont elles qui vont dicter les modifications de l'état des agents et de l'environnement.

Les communications explicites entre agents. Parmi ces actions figurent les communications entre agents, qui nécessitent la mise en place d'un protocole de communication lors de la modélisation pour garantir leur cohérence. L'envoi

[2] http ://ccl.northwestern.edu/netlogo/

de ces messages pose maintes problématiques, telles que la détermination du moyen utilisé pour communiquer, de la latence entre l'envoi du message et la réception de celui-ci, la gestion de collision des messages, la détermination de leur portée, la définition de la structure du langage [3], . . .Selon le positionnement vis à vis de ces problèmes, et l'implémentation qui en est faite, les résultats de la simulation aussi bien que la complexité du simulateur différeront.

Les communications simples entre agents. Des envois de messages simples sont souvent largement suffisants. Dans ce cas, il n'est pas forcément nécessaire de les prendre en compte à l'étape de modélisation car ils ne soulèvent pas ces difficultés

La stigmergie. La communication peut aussi avoir lieu au travers de l'environnement (communication stigmergique), dans quel cas il n'y a pas explicitement échange de messages entre les agents, la « communication » étant limitée à la production et la perception de ressources spécifiques de l'environnement.

Sa complexité est donc liée à la représentation des ressources de l'environnement et à la perception de celles-ci.

3.5 La sélection de comportements

La sélection de comportement se base sur la condition d'éxecution pour départager les divers comportement et en sélectionner un. Dans les cas les plus simples, ces conditions sont disjointes, donnant ainsi naissance à une sélection triviale. Des conditions non disjointes introduisent une ambiguïté qu'il est parfois nécessaire de supprimer, en mettant en œuvre des mécanismes plus ou moins évolués.

Les priorités. Une des architectures les plus couramment utilisées pour résoudre ce problème hiérarchise les comportements par priorités. La résolution des conflits est alors laissée au modélisateur, qui doit définir des priorités pour chaque comportement.

L'introduction de la hiérarchie dans les comportements simplifie la prise de décisions dans le cas d'ambiguïtés entre plusieurs comportements, puisque c'est à chaque fois le comportement réalisable de plus haute priorité qui sera sélectionné. Le problème est alors reporté dans les cas où plusieurs comportements réalisables disposent de la même priorité, pour lesquels aucune métrique n'est spécifiée. Pour ce genre de cas, l'ambiguïté est voulue par le modélisateur, et il est donc possible de sélectionner les comportement indifféremment, par exemple dans le cas de comportements à priorités égales ayant des conditions disjointes et dont les priorités se suivent.

L'exemple le plus courant d'un tel type d'architecture est l'architecture de *subsomption* proposée dans [1], utilisée implicitement par toute simulation dont la sélection d'action s'effectue avec des conditions imbriquées.

3.6 Conclusion

Les divers points présentés dans cette partie permettent d'avoir une idée des gains occasionnés par les choix de simplification effectués à la modélisation (perte de finesse dans le modèle), et les biais étant implicitement ajoutés.

La partie suivante expose, comme nous l'avons fait avec le modèle, en quoi l'implémentation augmentera ou diminuera la complexité du moteur de simulation. C'est seulement alors qu'il est possible d'exprimer concrètement la complexité du moteur de simulation.

4 La complexité logicielle

Bien que la modélisation du moteur de simulation définisse une partie de la complexité, c'est son implémentation qui la fixe *in fine*. Lors de l'implémentation, tout programmeur est tenté d'effectuer des optimisations pour rendre la simulation informatique plus performante, de réutiliser des outils déjà existants, sans réellement se soucier de l'effet que cela peut avoir sur les résultats, et introduisent parfois des biais. Cette partie a pour but d'exposer les choix les plus courants d'implémentation, et l'effet qu'ils ont sur la complexité et les résultats de la simulation.

4.1 Le temps

Le temps est une notion complexe à prendre en compte dans les simulations, et des approches de simplification sont introduites dès la modélisation. Les implémentations de celles-ci sont multiples, introduisant dans certains cas des biais risquant de changer l'interprétation possible des résultats.

L'approche Asynchrone. L'approche asynchrone utilise du temps continu, alors que les ressources informatiques utilisées ne proposent qu'une plateforme séquentielle. Il est donc nécessaire de simuler le temps continu, ce qui est un premier biais qui, malheureusement, ne peut être évité.

Une approche continue pure suppose que tout agent puisse effectuer des cycles *perception - sélection de comportement - exécution* parallèlement aux autres agents et à l'évolution de l'environnement. Chaque agent dispose d'un ensemble de processus permettant l'évolution continuelle de son état, le choix de ses actions en temps réel, et éventuellement pour gérer les messages provenant d'autres agents ou la perception de des changements dans l'environnement.

Il faut alors mettre en place des mécanismes de cohérence entre les divers processus, voire résoudre des problématiques proches des préoccupations dans le domaine du temps réel, telles que la prévision du temps de calcul requis par chaque processus, la gestion d'échéanciers pour maintenir la cohérence de l'état des agents et de l'environnement, du balancement de charge dans le cas d'architectures multi-processeurs...

Il est aisé de remarquer que cette approche est excessivement complexe, expliquant pourquoi elle est rarement utilisée. Certaines des problématiques présentées n'ont pas lieu dans le cas où le temps de simulation continu peut être du temps simulé (une seconde de temps simulé ne doit pas obligatoirement correspondre à une seconde du temps réel), mais le nombre conséquent de processus utilisés reste le même.

La plupart des simulations en temps continu préfèrent l'utilisation des événements discrets à l'utilisation du temps continu pur, car ils permettent de restreindre le nombre de processus requis par la simulation en utilisant une horloge discrète cadencée par l'occurence des événements. Le moteur de gestion d'événements effectue en boucle : avancer son horloge à l'occurrence de l'événement le plus proche, mettre à jour l'état des agents et de la partie de l'environnement concernée par le comportement associé à l'événement, et enfin exécuter le comportement, qui lui même peut générer d'autres événements. Il n'est alors plus nécessaire de maintenir les processus de mise à jour, réduisant ainsi les ressources utilisées.

L'approche Synchrone. A l'image des conflits de l'évolution des états dans l'approche asynchrone, l'approche synchrone pose le problème de la mise en commun des différents changements issus de l'ensemble des comportements exécutés à un même pas de temps. Ce problème est non trivial, car on ne connaît que l'état final dans lequel se trouve l'agent : les métriques se basant sur la façon d'évoluer de l'état ne sont plus applicables. La métrique la plus utilisée est la sélection d'un agent qui modifiera l'état, invalidant l'action de tous les autres.

Cette approche nécessite une grande quantité de ressources mémoire (chaque agent dispose d'une copie de l'environnement), ce qui la rend peu pratique à mettre en œuvre. Des approches dérivées existent, permettant d'approcher la simultanéité des actions sur un moteur de simulation séquentiel, par exemple en conservant un journal des états modifiés dans le pas de temps et résolvant les conflits à la fin du pas de temps pour les états modifiés plus d'une fois, en général de la même façon que dans le cas séquentiel (exemple : [2]).

Une seconde approche est d'évaluer l'effet des actions au travers d'un modèle à influences [5], pour lequel chaque état se voit associer un ensemble d'influences issues d'actions (à la place de modifications directes de l'état), qui décriront les tendances d'évolution de celui-ci. La difficulté réside alors dans le calcul de la réaction (i.e. comment les influences sont prises en compte dans la valeur de l'état à la fin du pas de temps).

L'approche Séquentielle. Une approche séquentielle pure prend en compte la résolution de conflits implicitement, introduisant des biais, mais éliminant du coup une des principales sources de la complexité dans la simulation. Tout du moins, pour éviter de favoriser certains agents par rapport à d'autres, il est nécessaire de choisir équitablement à chaque pas de temps l'ordre dans lequel les agents sont considérés, ce qui implique souvent l'utilisation de générateur de nombres aléatoires, et pose donc le problème de leur complexité intrinsèque et les difficultés dans leur conception [9].

4.2 L'Environnement

La gestion de l'espace. La complexité de la mesure du voisinage en environnement continu est issue de la répétition du test d'appartenance au voisinage itéré sur l'ensemble des agents de la simulation. Pour réduire cette complexité, il est possible d'utiliser un environnement continu par

morceaux. La recherche des voisins se restreint alors aux cellules proches, réduisant le nombre total d'agents à tester (voir figure 3).

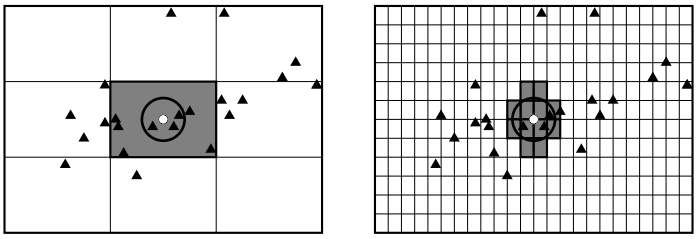

FIG. 3 – Approximations de l'environnement continu. Le cercle correspond au champ visuel de l'agent, les triangles correpondent aux autres agents dans la simulation, et les cases grises correspondent aux cases voisines (le champ visuel de l'agent doit être inclus dans l'ensemble des cases voisines sélectionnées).

La complexité de perception du voisinage se répartit sur les opérations de recherches de cellules voisines et de recherche des agents voisins dans ces cellules. Plus la discrétisation de l'environnement augmente (i.e. plus la taille des cellules est faible), plus la recherche des cellules voisines risque d'être complexe, et plus la recherche de voisins dans ces cellules sera simple.

4.3 Les comportements

Les communications non-stigmergiques introduisent la notion de message passant d'un agent à un autre. Il est alors nécessaire de définir les moyens utilisés pour transférer ces messages entre les agents, et ainsi définir si les envois de messages sont synchrones (un agent attendant une réponse à un message ne peut pas effectuer d'autres actions) ou asynchrones (les agents décident quand vérifier s'ils n'ont pas reçus de messages), si les communications se font en mode connecté ou pas (i.e. si un protocole de communication est initié entre deux agents, peut-il être perturbé par un agent tiers), si la communication se fait point à point (d'un agent source à un agent cible sans que les autres agents en aient connaissance), en diffusion globale ("broadcast" du message à l'ensemble des agents) ou locale ("multicast" du message limité aux agents du voisinage), définir comment détecter et gérer les collisions ou pertes de messages (un message peut être diffusé vers un agent qui s'est déplacé et n'est plus à portée de communication).

Tous ces problèmes sont largement traités par le domaine des réseaux informatiques, proposant pour cela des mécanismes plus ou moins évolués et donc plus ou moins complexes.

Il est possible de limiter les communications a des appels de méthodes sur les agents destinataires du message ou sur l'environnement. Ce gain de simplicité est acquis au prix de biais : il supprime l'éventuelle notion temporelle du message (latence entre envoi et réception).

4.4 La sélection de comportements

La sélection de comportements par priorités constitue une optimisation de la sélection classique, puisqu'elle propose d'effectuer le test sur des sous-ensembles de comportements itérativement par priorités décroissantes plutôt que directement sur la totalité des comportements possibles : dès qu'un comportement est déterminé réalisable à une priorité p donnée, les priorités inférieures ne seront pas parcourues. Le choix d'un comportement est alors restreint aux comportements réalisables de la priorité p. Ce gain est non négligeable, puisque le nombre de tests effecutés pour chaque comportement est fonction du nombre d'agents voisins et de la cardinalité du comportement. Par exemple, pour 3 comportements a, b et c, avec 5 voisins et ayant pour cardinalité respectives 1, 2 et 3 (arrangements), le nombre de tests de condition d'exécution à effectuer dans le cas classique est de 85. Si b et c sont de même priorité, et que a est la priorité maximale, ce nombre diminue à 5 s'il existe un voisin pour lequel a est réalisable, à 85 sinon (il faut tester a, puis b et c).

Si les comportements sont à priorités décroissantes, les tests requis sont de 5 si a est réalisable pour un des voisins, 25 si a mais b oui, et 85 sinon. Ce gain est donc non-négligeable, d'autant plus si le nombre de voisins et la cardinalité des comportements sont importants. Ainsi, une chambre à bulle, n'impliquant que peu d'agents et un seul comportement, sera moins complexe de ce point de vue qu'une simulation de type Age of empires pour laquelle les agents et les comportements sont nombreux.

4.5 Vers une conception orientée interactions

Comme nous l'avons montré, une façon de permettre une évaluation fine de la complexité de la simulation commence par la dissociation entre moteur de simulation, agents et comportements. Nous avons proposé dans ce sens une approche

consistant à définir, dès la phase de conception, des interactions indépendantes logiciellement des agents. Ces interactions sont ensuite affectées à des agents pouvant jouer le rôle de source ou de cible (cf § 2 p 2). Cette analyse impose d'établir explicitement une matrice d'interaction [10] qui donne une première estimation de la complexité du modèle.

Nous avons également montré comment donner à un tel genre de simulation une implémentation comportant aussi peu de biais que possible, dans le cadre d'interactions entre une source et une cible. Il s'agit là d'une première étape en vue de la définition de simulations plus générales faisant intervenir une complexité arbitraire des modèles de comportement et d'environnement.

5 Conclusion

Nous avons identifié ci-dessus un certain nombre de choix d'implémentation cruciaux qui permettent de discriminer les plateformes de simulation multi-agent selon les choix qu'elles font : il s'agit en particulier de la distinction entre espace discret ou continu, de l'ordonnancement des actions des agents ou de la représentation de l'environnement soit par des agents, soit par des artefacts dédiés.

Afin de déterminer la complexité d'un modèle de comportement donné et de son implémentation, nous avons d'abord évalué la complexité algorithmique inhérente au processus de simulation en général, où le coût majeur provient de la détection des voisins et des interactions réalisables entre agents.

Nous avons ensuite montré l'impact des choix d'implémentation sur la complexité des modèles simulables par ces plateformes : ainsi par exemple, les modèles à événements asynchrones doivent en pratique être réalisés sur des machines à temps discret, ce qui entraîne une réduction forte de la complexité du modèle d'origine. Ignorer ce décalage entre modèle conceptuel et modèle opérationnel, c'est introduire des biais dans le déroulement de la simulation : le résultat de simulation ne reflète pas ce qu'on peut attendre du modèle conceptuel. De même certains choix d'implémentation se traduisent par une augmentation indue du temps de calcul.

Pour améliorer la conception et la réalisation des simulations multi-agents, il est donc crucial de séparer radicalement la part de complexité inhérente à la simulation (par exemple au moyen d'un formalisme orienté interactions comme dans [10]) de celle qui dépend du domaine étudié (le modèle conceptuel) ou du code (le modèle opérationnel). Ce n'est qu'à cette condition qu'on peut établir, en connaissance de cause, un compromis entre la complexité d'un modèle et sa réalisation plus ou moins biaisée sur une plateforme de simulation.

Références

[1] Brooks, R. A., and J. Connell, *Asynchronous Distributed Control System for a Mobile Robot*, Proceedings of SPIE's Cambridge Symposium on Optical and Optoelectronic Engineering, Cambridge, MA, October 1986, p 77-84. thèse de Fabien Michel à propos de l'architecture de subsomption)

[2] Campos A. M. C., *Une architecture logicielle pour le développement de simulations visuelles et interactives individu-centrées : application à la simulation d'écosystèmes et à la simulation sur le Web*, Thèse de Doctorat, Université Blaise Pascal, Clermont II, 2000

[3] Demazeau Y., *From Interactions to Collective Behaviour in Agent-Based Systems.* In Proceedings of the 1st European Conference on Cognitive Science, Saint-Malo, 1995

[4] Edmonds B., *Simulation and Complexity-how they can relate*, Virtual Worlds of Precision, Valerie Feldmann, Katrin Mühlfeld (eds.), 2005

[5] Ferber J. and Müller J.P., *Influences and Reaction : a Model of Situated Multi-agent Systems*, Tokoro, Mario (ed), Proceedings of the 2nd International Conference on Multi-agent Systems (ICMAS-96). The AAAI Press, p 72-79, 1996

[6] Genesereth M. R. and Nilsson N. J., *Logical foundations of artificial intelligence*, Morgan Kaufmann Publishers Inc., 1987

[7] Gershenson C., *When can we call a system self-organizing ?*, Advances in Artificial Life, 7th European Conference, Dortmund, Germany, 2003

[8] Law A.M., *Simulation model's level of detail determines effectiveness*, Industrial Engineering 23 (10), p 16-18, 1991

[9] L'Ecuyer P., *Random numbers for simulation*, Communications of the ACM, Volume 33 issue 10, p 85-97, 1990

[10] Mathieu P., Picault S., Routier JC., *Donner corps aux interactions (l'interaction enfin concrétisée)*, 4e journees francophones sur les modeles formels d'interaction (MFI-2007), Paris, 2007

[11] Meurisse T., *Simulation multi-agent : du modèle à l'opérationnalisation*, Thèse de Doctorat, Université Pierre et Marie Curie-Paris VI, 2004

[12] Michel F., *Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents*, PhD thesis - Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier - Université Montpellier II, 2004

[13] Schelling T.C., *Micromotives and Macrobeahvior*, W. W. Norton and Co., p 147-155, 1978