Chapitre 6

Résumé des résultats obtenus

6.1 Systèmes Multi-Agents

Ces résultats proviennent pour l'essentiel de mon travail de thèse.

6.1.1 Introduction

L'évolution de la programmation vers des composants de plus en plus puissants et indépendants a conduit à l'émergence des Systèmes Multi-Agents (SMA). Ils ont aussi conquis une place importante dans les systèmes de prise de décision où ils remplacent les humains, notamment en commerce électronique. Ma thèse ce concentre sur ce deuxième domaine, mais de nombreux résultats s'appliquent au premier.
Dans un contexte de forte concurrence économique, les agents doivent être considérés comme totalement autonomes, c'est-à-dire capable de prendre n'importe quelle décision à n'importe quel instant. La difficulté est alors de parvenir à faire en sorte que ces agent atteignent un consensus. C'est le problème que nous avons attaqué, mais il nécessite une longue approche.

6.1.2 Autonomie

Bien qu'étant un concept central dans la définition d'un agent, l'autonomie a longtemps été définie de façon floue. Elle a souvent été considérée plus comme un fil conducteur que comme un contrainte forte, ce qui a amené une forte divergence dans la conception même de la notion de SMA. Pourtant, l'autonomie est fondamentale pour permettre une adaptation dynamique des agents au contexte, en leur laissant toute liberté de prise de décision.
Nous distinguons trois niveaux d'autonomie.
Au niveau agent, il existe aujourd'hui plusieurs définitions de l'autonomie, mais beaucoup sont en fait des autonomies partielles, se basant sur l'argument que l'autonomie totale ne permet pas de contrôle du système. En fait, le comportement des agents n'est pas prédictible, parce qu'ils sont hétérogènes et ont leur propre stratégie, parce qu'ils peuvent tricher, et parce qu'ils peuvent avoir des fautes matérielles ou logicielles. La question est alors : comment contrôler le comportement d'un agent totalement autonome ? Pour cela, il faut que les contraintes spécifiées par les protocoles ne reposent pas sur le processus de décision interne qui est caché, mais sur les données observables, c'est-à-dire les messages échangés. On peut alors contrôler ces messages échangés pour vérifier qu'ils respectent bien le protocole, et en cas de fraude, une sanction doit être appliquée. Cette sanction est basée sur la rationalité de l'agent (ex : une amende pour une rationalité économique).
Au niveau organisationnel, les agents doivent avoir la liberté de choisir leurs interactions : période, partenaires, cause. Les alliances répondent à ces contraintes : ce sont des groupes d'agents qui se forment et se défont dynamiquement en fonction des besoins.
Au niveau système, les agents devraient aussi avoir la liberté de choisir leur protocoles, de les créer, de les adapter. Mais ces modification doivent se faire selon une certaine règle, car changer un protocole peut avoir des conséquences sur l'issue des interactions. Si l'on compare les protocoles de niveau habituels à des lois, cela revient à dire qu'il faut une constitution. Nous n'attaqueront pas ce problème dans cette thèse.

6.1.3 Consensus

Dans un système d'agent totallement autonomes, une prise de décision qui touche plusieurs agents ne peut se faire que par consensus. Pour respecter l'hétérogénéité des agents, nous avons cherché un protocole qui ne nécessite pas d'importantes capacités de calcul, ni de compétences particulières (cognitives par exemple).
Pour atteindre un consensus, les agents doivent s'échanger de l'information pour tenter de modifier les opinions des autres. L'information la plus simple à échanger est la position sur laquelle ils sont actuellement.
Afin d'inciter les agents à faire évoluer leur position, un blocage déclenche une procédure qui fait fusionner des agents au sein de coalitions.
Le protocole utilise en outre une diffusion parallèle cryptée pour s'assurer que tous ont accès à l'information en même temps. La complexité de l'algorithme est exponentielle dans le pire des cas, mais les expérimentations montrent que la convergence est assez rapide en pratique.

6.1.4 Opinion

Notre protocole nécessite un formalisme de représentation d'opinions, vus comme des ensembles de préférences, et de plusieurs opérateurs. Tous les formalismes proposés contiennent une part plus ou moins importante de transitivité. Pour nous, une préférence est une notion locale, de comparaison entre deux options, qui ne doit pas faire référence aux autres préférences.
Nous avons donc proposé un formalisme original de représentation des opinions qui permet de plus de calculer de façon très naturelle l'opinion d'un groupe, nécessaire à notre protocole.
Les expérimentations montrent que la satisfaction à une allure gaussienne (peu de mécontents et de très satisfaits, beaucoup de moyennement satisfaits), alors que le système de vote produit une satisfaction à allure uniforme (tous les degrés de satisfaction sont à peu près représentés avec le même poids).

6.1.5 Alliance

L'autonomie au niveau organisationnel nous oblige à permettre aux agents de s'organiser selon des structures souples et dynamiques, les alliances.
Le problème de la formation d'alliance entre agents totalement autonomes peut être résolu en le considérant comme un problème d'atteinte de consensus sur la structure organisationnelle à atteindre.

6.1.6 Implementation

Afin de tester les protocoles, j'ai développé une plateforme qui permet de les tester rapidement, en facilitant leur implémentation. Le code source pèse 145Ko pour la partie agent et opinion, et 32Ko pour la partie Réseau de Petri qui permet la simulation des protocoles.

6.1.7 Expérimentations

Pour expérimenter notre protocole, j'ai effectuer des tournois entre différentes stratégies, en faisant varier les paramètres.
Les résultats montrent que les agent les plus rigides ne sont pas gagnants, ce qui tend à les rendre plus flexibles, et qu'une forte compétition conduit à un consensus plus rapide.

6.1.8 Conclusion

Dans un contexte d'agents totalement autonomes, nous avons proposé un protocole dont les règles sont basées sur de données observables, qui respectent l'autonomie et l'hétérogénéité et dont les utilisateurs

sont motivés selon leur rationalité. Les expérimentations montrent que les stratégies les plus néfastes ne sont pas payantes.

6.2 Traitement Automatique de la Langue

Ces résultats proviennent de la recherche que j'ai effectuée lors de mon Post-Doc sur le projet ALVIS. Voir `http://www-src.lip6.fr/homepages/Guillaume.Vauvert/alvis/alvis_format.pdf` pour plus de détails.
J'ai proposé une architecture de traitement qui encapsule chaque outil existant dans des traducteurs afin qu'il n'existe qu'un seul format d'échange entre outils.
Après avoir étudié en profondeur de nombreux formats d'annotation de document, notre choix s'est porté sur la création d'un nouveau format en XML, qui permette une représentation fine des information lingusitiques, et qui autorisent les structures non-arborescentes. De plus, à travers ce format, nous avons dû faire un choix de la plus petite unité représentable.
À travers ce Post-Doc, j'ai découvert XML et tous les outils atenants. J'ai aussi plongé dans les outils de Traitement Automatique de la Langue, renouant ainsi avec mon stage de DEA.
J'en sors convaincu que beaucoup de difficultés seraient résolues par une approche distribuée, par les Systèmes Multi-Agents.

6.3 Systèmes Multi-Agents large échelle

Ces résultats sont issus de la recherche que je mène actuellement au sein du projet DARX. Voir `http://www-src.lip6.fr/homepages/Guillaume.Vauvert/inria_report.pdf` pour plus de détails. Il s'agit d'un travail en cours, mais déjà des résultats intéressants ont vu le jour.

DIMAX est une plateforme pour Système Multi-Agents, qui tourne au-dessus de DARX, un middleware qui autorise la réplication des processus selon plusieurs stratégies. La puissance de calcul étant de plus en plus disponible sous la forme de machines en réseau (grappe, grille), les applications réparties large échelle – notamment les Systèmes Multi-Agents large échelle – utilisent un grand nombre de ressources sur des durées telles que la probabilité de panne est non négligeable.
Il est donc nécessaire d'anticiper la panne par la réplication des processus, la seule réponse fiable. Étant impossible de répliquer tous les agents, il est nécessaire de choisir quels sont les plus critiques. Les approches existantes consistaient à transmettre des indices de criticité de la couche SMA à la couche système, qui devait alors décider qui et comment répliquer les agents. Estimant qu'il s'agissait d'un goulet informationnel, j'ai cherché à définir plus précisément la criticité. J'en suis arrivé à la considérer comme les conséquences d'une faute en terme de Qualité de Service (QoS); la probabilité de faute est aussi prise en compte. Ce modèle est très général, mais prédire les conséquences de la faute est très dépendant de l'application. Un vaste chantier s'ouvre pour trouver des solutions réutilisables pour permettre à des applications large échelle de fonctionner le mieux possible du point de vue de l'utilisateur (la QoS du système), en prenant en compte ces possibilités de faute. Il semble nécessaire d'inclure dans le système des connaissances du concepteur de l'application, des préférences de l'utilisateur de l'application, des connaissances dynamiques de l'application (par observation en cours d'exécution), des connaissances dynamiques sur l'utilisation des ressources et d'autres connaissances. L'objectif est alors de permettre l'intégration de toutes ces connaissances pour prendre la meilleure décision.
Cette approche semble des plus prometteuses de par sa généralité, mais elle nécessite encore de la recherche pour être opérationnelle. L'utilisation d'agents pour superviser l'exécution de l'application permet une flexibilité et une richesse d'échange informationnelle qui devraient donner des meilleurs résultats que l'approche classique.