textOCampus

Diffusion adaptative et massive de SMS

Rapport de Master 2 Recherche
Intelligence Artificielle, Intelligence Collective, Interaction (IAICI)

Université Paul Sabatier (UPS) – Toulouse III

07 Septembre 2015

Elhadi Belghache

Laboratoire d'accueil : Institut de Recherche en Informatique de Toulouse (IRIT)
Directeur de recherche : Marie-Pierre Gleizes
Responsable de stage : Pierre Glize
Equipe d'accueil : Systèmes Multi-Agent Coopératifs (SMAC)

Mots-clés : SMA, adaptation, émergence, diffusion d'information, réseau de communication

---

Résumé : L'ensemble des laboratoires de recherche de Paul Sabatier impliqués dans l'opération neOCampus œuvrent en synergie pour améliorer le confort quotidien sur le campus tout en diminuant l'empreinte écologique des bâtiments et en réduisant les coûts de fonctionnement (fluide, eau, électricité...). Dans ce cadre nous souhaitons mettre à disposition un système d'échange massif d'informations par SMS qui diffuse spontanément à un groupe d'usagers une information émise par l'un d'eux. Conformément aux contraintes édictées dans neOCampus, aucun humain ne devrait intervenir pour décider des destinataires pertinents d'un message : textOCampus doit apprendre en autonomie complète et en temps réel.
textOCampus emploiera la technologie des systèmes multi-agent auto-adaptatifs pour ce faire. Il sera constitué de dizaines de milliers d'agents interdépendants réalisant une activité collective bien plus élaborée que leurs capacités individuelles. L'adaptation du collectif provient de l'aptitude des composants à s'auto-organiser durant leur activité.

Remerciements

À M. Pierre Glize, Mme Marie-Pierre Gleizes et M. Frédéric Amblad,

À tous les enseignants qui m'ont permis d'en arriver là,

À toutes les personnes qui ont contribué, de près ou de loin, à ce projet,

À toute ma famille, à tous mes proches, qui m'ont soutenu et encouragé,

Merci

Table des matières

INTRODUCTION **1**

1 ETAT DE L'ART : DETECTION DE COMMUNAUTES **3**
1.1 CONCEPTS DE BASE 3
1.1.1 Communauté 3
1.1.2 Chevauchement 4
1.1.3 Hiérarchie 5
1.1.4 Partition et couverture 5
1.2 TECHNIQUES DE DETECTION DE COMMUNAUTES 6
1.2.1 Optimisation de la modularité 6
1.2.1.1 Modularité 6
1.2.1.2 Maximisation de la modularité 7
1.2.1.3 Limite de la maximisation de la modularité 7
1.2.1.4 Multirésolution 8
1.2.2 Autres techniques 8
1.2.2.1 Partitionnement algorithmique 9
1.2.2.2 Optimisation multiobjectif 9
1.2.3 Détection de communautés dynamiques 9
1.2.4 Simulation à base d'un Système Multi-Agent 11
1.3 BILAN 11

2 CONTEXTE DE L'ETUDE **12**
2.1 SYSTEMES MULTI-AGENT 12
2.1.1 Introduction 12
2.1.2 Concepts associés aux SMA 12
2.1.2.1 Agent 12
2.1.2.2 Environnement 13
2.1.2.3 Interactions 14
2.1.2.4 Organisations 14
2.1.3 Emergence et auto-organisation 14
2.2 THEORIE DES AMAS 15
2.2.1 Adéquation fonctionnelle des AMAS 16
2.2.2 Processus d'apprentissage par auto-organisation 16

2.2.3 Situations Non Coopératives (SNC) 17
2.3 METHODE ADELFE 17
2.4 BILAN 18

3 TEXTOCAMPUS 19
3.1 BESOINS PRELIMINAIRES 19
3.1.1 Objectifs 19
3.1.1.1 Objectif général 19
3.1.1.2 Objectif spécifique au domaine 20
3.1.2 Besoins 20
3.1.2.1 Besoins généraux 20
3.1.2.2 Besoins spécifiques au domaine 20
3.1.3 Critères nécessaires à l'évaluation 20
3.1.3.1 Critères généraux 20
3.1.3.2 Critères spécifiques au domaine 21
3.1.4 Expérimentations 21
3.1.5 Mots-Clés 21
3.2 BESOINS FINALS 22
3.2.1 Environnement 22
3.2.2 Entités 22
3.2.3 Contexte 22
3.2.3.1 Flots de données entre les entités passives et le système 22
3.2.3.2 Interactions entre les entités actives et le système 22
3.2.4 Caractérisation de l'environnement 23
3.3 ANALYSE 23
3.3.1 Vérifier l'adéquation des AMAS 23
3.3.1.1 Vérifier l'adéquation au niveau global 23
3.3.1.2 Vérifier l'adéquation au niveau local 24
3.3.2 Identifier les agents 24
3.3.3 Interactions 24
3.3.3.1 Envoi de messages 24
3.3.3.2 Diffusion de messages 25
3.3.3.3 Vérification de diffusion 27
3.4 CONCEPTION 29
3.4.1 Identifier les agents 29
3.4.2 Situations Non Coopératives (SNC) 29
3.5 BILAN 31

4 EXPERIMENTATIONS ET ANALYSES 32
4.1 GENERATEUR DE SCENARIOS 32
4.2 EXPERIMENTATIONS ET ANALYSES 33

5 CONCLUSION 36

BIBLIOGRAPHIE 37

Table des illustrations

Figure 1.1 — graphe avec huit communautés 4
Figure 1.2 — Chevauchement (overlap) 4
Figure 1.3 — Hiérarchie de communautés 5
Figure 2.1 — graphe clairsemé (a) et graphe dense (b) 6
Figure 2.2 — évolutions possibles d'une communauté 10
Figure 2.1 — Adaptation par auto-organisation 16
Figure 2.1 — Diagramme de collaboration 22
Figure 2.2 — Diagramme de cas d'utilisation 23
Figure 3.1 — Diagramme de séquence pour l'envoi d'un message 25
Figure 3.2 — Diagramme de séquence pour la diffusion d'un message 27
Figure 3.3 — Diagramme de séquence pour la vérification de la diffusion d'un message 28

Introduction

Chaque jour, des dizaines de milliers d'usagers (étudiants, enseignants, personnels administratifs et techniques), envoient des informations sur leurs activités dans le campus en s'échangeant des textos (SMS) par le biais de leurs smartphones. Ces échanges entre individus forment un réseau social constitué de communautés ou groupes implicites, c'est-à-dire des ensembles de personnes ayant en commun des centres d'intérêt éducatifs, professionnels, sociaux ou autres.

En fonction de ses échanges, un individu peut appartenir, implicitement, à un ou plusieurs groupes (overlap) de façon ponctuelle ou plus régulière, car les centres d'intérêt d'un individu peuvent évoluer, ce qui induit une difficulté à cerner les différents groupes d'intérêt du réseau. D'autre part, l'énorme flot de textos circulant à travers le réseau lui confère une forte dynamicité, évoluant au fil de l'apparition, de la disparition des usagers et du changement des groupes constitutifs.

Alors, en dépit de ces complications, comment faire une synergie de toutes les informations échangées localement pour améliorer au mieux le fonctionnement global ? C'est-à-dire comment faire bénéficier tous les usagers des échanges de messages le plus efficacement possible ?

Ce projet a pour objectif d'y répondre en diffusant les messages à toutes les personnes intéressées par le sujet grâce à notre solution « textOCampus », un système d'échange massif d'informations par SMS qui diffuse spontanément à un groupe d'usagers une information émise par l'un d'entre eux. Ce système devra apprendre à cerner les groupes d'intérêt dynamiques en autonomie complète et en temps réel. Pour ce faire, textOCampus emploiera la technologie des systèmes multi-agent auto-adaptatifs (AMAS). Il sera constitué de dizaines de milliers d'agents interdépendants réalisant une activité collective bien plus élaborée que leurs capacités individuelles. L'adaptation du collectif provient de l'aptitude des composants à s'auto-organiser durant leur activité [Georgé et al. 2011].

Le stage fera donc l'évaluation sur des scénarios synthétiques de textos simulant des milliers d'envois de textos quotidiens entre plusieurs milliers d'usagers disposant de smartphones. Le développement de l'application portera donc sur deux points principaux :

1) un générateur de scénarios simulant des milliers de textos.

2) le système multi-agent adaptatif qui apprendra à diffuser de manière la plus ciblée possible les textos par observation des informations recueillies sur les smartphones.

Dans ce rapport, nous dresserons, en premier lieu, un bref état de l'art décrivant quelques techniques jusqu'alors utilisées pour résoudre des problèmes semblables au nôtre, puis nous présenterons le contexte de notre étude c'est-à-dire les définitions et concepts généraux des systèmes multi-agent et, plus particulièrement, ceux sur lesquels vont porter cette étude : les systèmes multi-agent adaptatifs (AMAS).

Nous donnerons également un aperçu des outils pouvant aider un concepteur de SMA en se focalisant plus spécifiquement sur la méthode de conception ADELFE, dédiée aux AMAS. En suivant cette méthode, nous décrirons alors l'adaptation des concepts de la théorie AMAS à la problématique de diffusion spontanée d'informations afin de construire notre solution, puis une analyse des performances de notre solution sera présentée au travers de résultats expérimentaux.

Nous conclurons ce document par un bilan du travail réalisé et la proposition de perspectives d'améliorations de notre solution.

1 Etat de l’art : Détection de communautés

L’élément clé pour une bonne diffusion de l’information dans un réseau social, sans connaissance explicite de l’appartenance des individus à des groupes d’intérêt (communautés), réside dans le fait de détecter avec précision les communautés implicites du réseau des utilisateurs afin de transmettre chaque message aux usagers les plus pertinents et à ceux-là uniquement.

La détection de communauté est devenue, ces dernières années, un problème de plus en plus important recevant une attention grandissante des scientifiques compte tenu des débouchés pratiques considérables qui en découlent.

En effet, la détection de communauté nous permet de comprendre et d’explorer un réseau et prédire des liaisons non observées. Par exemple, en cernant les communautés dans un réseau social [Newman et Girvan 2004], on peut déduire l’ensemble des personnes pouvant être amies avec un individu donné et qui ne sont pas encore reliées. Un autre exemple est de faire des hypothèses sur les domaines de recherche et leurs sous-domaines dans un réseau de citation d’articles scientifiques [Chen et Redner 2010].

1.1 Concepts de base

La détection de communauté est apparue sous des formes variées dans plusieurs disciplines comme la sociologie (analyse des réseaux sociaux), la biologie et l’informatique, disciplines où des systèmes sont souvent représentés comme des graphes (ensembles de sommets reliés par des arêtes). C’est pourquoi, les principes et concepts de base liés à la détection de communautés se rapportent à la théorie des graphes.

1.1.1 Communauté

Le terme « communauté » renvoie à un contexte social et la société offre une grande variété d’organisation de groupe : familles, amis, collègues, villes, pays, etc. En outre, avec la diffusion d’Internet des groupes appelés « communautés virtuelles » ont vu le jour.

Communément, les membres d’une même communauté partagent des centres d’intérêt communs et d’une manière plus générale, les entités constituant une communauté partagent les mêmes

propriétés spécifiques au domaine, comme par exemple dans le World Wide Web une communauté correspond à un ensemble de pages traitant le même sujet ou des sujets similaires.

Ainsi donc bien qu'il n'y ait pas de consensus sur la définition exacte d'une communauté, la majorité des scientifiques s'accordent à dire qu'une communauté ou un cluster dans un graphe est un ensemble de sommets (nœuds) reliés plus entre eux qu'au reste du graphe, en d'autres termes une communauté a plus d'arêtes (liens) à l'intérieur qu'à l'extérieur. De ce fait, une communauté apparaît dans un graphe comme une région avec une grande densité de liens internes, plus grande que la densité moyenne de liens dans le graphe comme le montre la Figure 1.1.1.

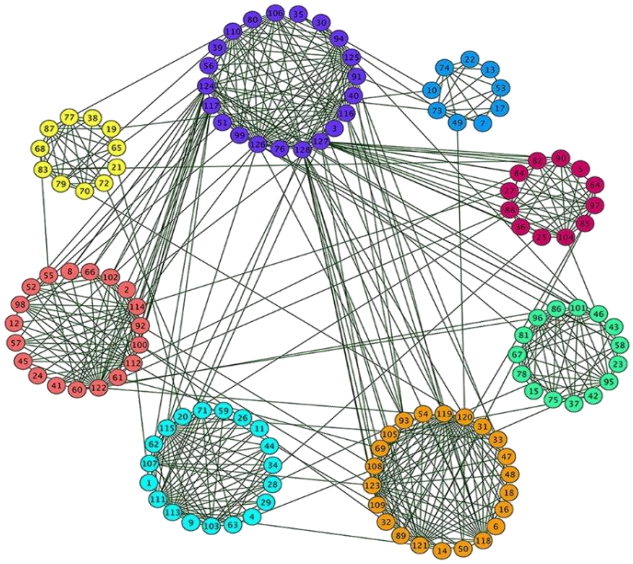
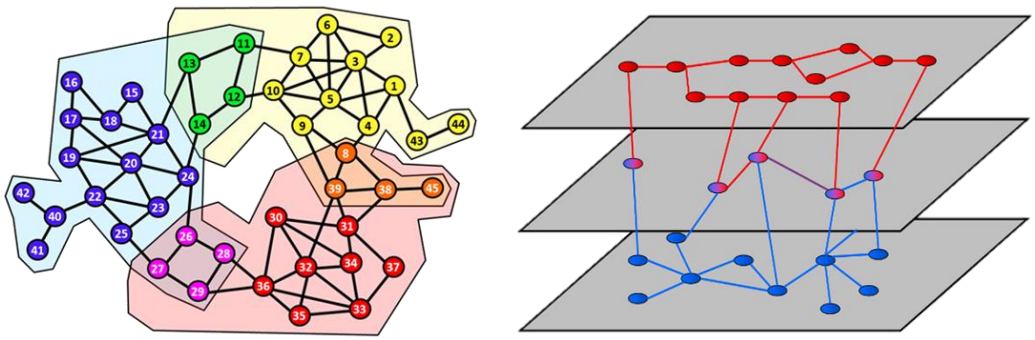


Figure 1.1.1 — graphe avec huit communautés

1.1.2 Chevauchement

Dans beaucoup de situations, il est naturel de penser qu'un nœud dans un graphe peut appartenir non pas à une unique communauté mais à plusieurs en même temps (voir Figure 1.1.2).



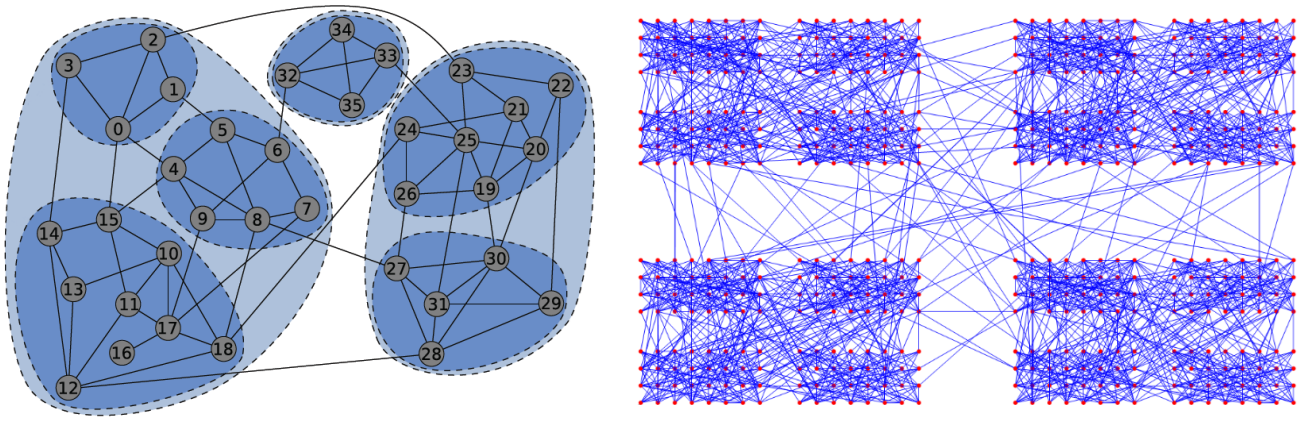
(a) **(b)**

Figure 1.1.2 — Chevauchement (overlap)

Un individu dans un réseau social peut ainsi appartenir à la fois aux communautés « école », « famille » et « club de karaté ». Par exemple, un article portant sur l'algorithme de Shor peut être cité en physique théorique, en informatique quantique et en sécurité informatique.

1.1.3 Hiérarchie

Un autre aspect important lié à la structure de communauté est l'organisation hiérarchique présente dans la plupart des systèmes en réseau du monde réel, car beaucoup de communautés sont imbriquées les unes dans les autres : de petites communautés forment de plus grandes, lesquelles se regroupent pour composer encore de plus grandes communautés comme on peut le constater sur la Figure 1.1.3.



(a) **(b)**

Figure 1.1.3 — Hiérarchie de communautés

En présence de hiérarchie, le concept de communauté devient plus riche et demande des techniques capables de détecter non pas un seul mais tous les niveaux de communautés.

Le corps humain est un exemple parfait d'organisation hiérarchique, puisqu'il est composé d'organes, lesquels sont composés de tissus et les tissus de cellules, etc. Un second exemple pourrait être l'organisation d'un réseau social d'enfants d'une même ville, on pourrait grouper les enfants selon l'école à laquelle ils vont, puis dans chaque école on peut les subdiviser en classes.

1.1.4 Partition et couverture

Une partition d'un graphe est une division du graphe en cluster tel que chaque sommet appartient à un cluster. Le nombre total de partitions d'un graphe grandit exponentiellement avec la taille du graphe (le nombre de sommets), ce qui signifie que l'énumération et/ou l'évaluation de toutes les partitions d'un graphe est impossible, sauf si le graphe est d'une petite taille.

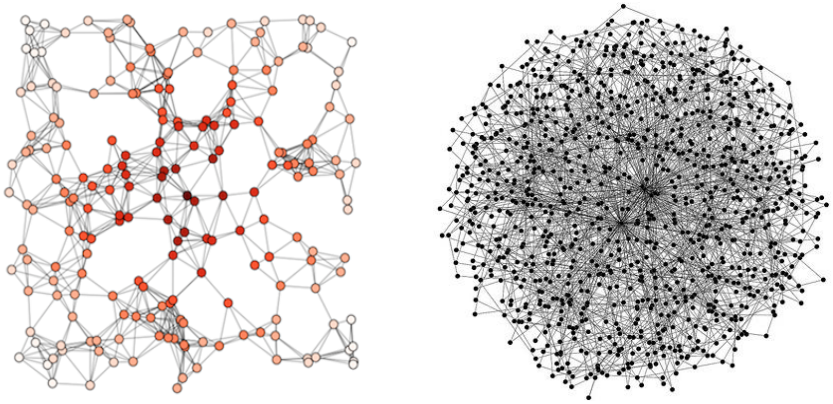
De plus, les partitions peuvent être hiérarchisées lorsque le graphe est lui-même hiérarchisé, augmentant donc le nombre de partitions possibles.

Comme vu précédemment (1.1.2), un sommet peut appartenir à plusieurs communautés. Une division d'un graphe en clusters qui se chevauchent est appelée une couverture. Prendre en compte de tels chevauchements introduit une variable supplémentaire, l'appartenance des sommets à différentes communautés, ce qui une fois de plus augmente le nombre de divisions possibles pour un graphe. En conséquence, la recherche de communautés avec chevauchement est plus coûteuse que la détection de simples partitions.

1.2 Techniques de détection de communautés

La détection de communauté est un champ de recherche actif donnant naissance à une vaste gamme de techniques inspirées des concepts et des méthodes de l'informatique, des mathématiques discrètes, de la sociologie et aussi de la physique statistique. Cela n'est pas surprenant puisque le concept de communauté n'est pas rigoureusement défini et nécessitent un certain caractère arbitraire et/ou de sens commun.

Il est important de souligner le fait que la détection de communautés est possible seulement si le graphe est clairsemé (voir Figure 1.2.1(a)), c'est-à-dire le nombre d'arêtes L est de l'ordre du nombre de sommets N. Et si $L \gg N$, la distribution des arêtes entre les sommets est trop homogène pour que des communautés prennent sens (voir Figure 1.2.1(b)). Dans ce cas, le problème devient assez différent, proche de la classification de données.



(a) **(b)**

Figure 1.2.1 — graphe clairsemé (a) et graphe dense (b)

1.2.1 Optimisation de la modularité

D'après les concepts définis jusqu'ici, on peut déduire que la détection de communautés revient à trouver de bonnes partitions d'un graphe. Mais qu'est-ce qu'une bonne ou une mauvaise partition ? Ou encore, comment mesure-t-on la qualité d'une partition ?

Il est par conséquent utile voire nécessaire d'avoir une fonction de qualité qui assigne à chaque partition d'un graphe un score représentant sa qualité. Ainsi, on peut classer les partitions sur la base de ce score et donc la meilleure partition est celle avec le plus grand score.

1.2.1.1 Modularité

La fonction de qualité la plus utilisée est la modularité de Newman et Girvan [Newman et Girvan 2004], elle est basée sur l'idée qu'un graphe aléatoire n'est pas supposé avoir de communautés :

$$Q = \sum_{s=1}^{m} \left[\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right],$$

Où m est le nombre de sous-graphe de la partition, l_s est le nombre de liens (arêtes) à l'intérieur du sous-graphe s, L est le nombre total des liens du graphe, et d_s est la somme des degrés des nœuds (sommets) du sous-graphe. le premier terme de la somme est la fraction des liens internes du sous-graphe s (densité des liens de s) ; le second terme représente la fraction attendue dans le sous-graphe si les sommets du graphe étaient reliés aléatoirement sans former de communautés (graphe aléatoire ou modèle nul).

Si pour un sous-graphe, le premier terme est beaucoup plus grand que le second, cela signifie qu'il y a plus de liens à l'intérieur que ce que l'on pourrait espérer avoir par hasard, cela revient à dire que le sous-graphe est une communauté. On en conclut que le sous-graphe s est une communauté si :

$$\frac{l_s}{L} - \left(\frac{d_s}{2L}\right)^2 > 0.$$

1.2.1.2 Maximisation de la modularité

De grandes valeurs de la modularité indiquent de bonnes partitions. Ainsi, la partition avec la valeur maximale doit être la meilleure ou une très bonne tout du moins. C'est la raison principale pour motiver une maximisation de la modularité Q par la plupart et les plus connues des méthodes de détection de communautés.

Une optimisation exhaustive de la modularité est impossible à cause de l'énorme nombre de divisions (partitions ou couvertures) possible pour un graphe même si celui-ci est petit. En outre, le maximum (optimum global) est hors d'atteinte et il a été prouvé que l'optimisation de la modularité est un problème NP-Complet [Brandes et al. 2006], donc il est probablement impossible de trouver la solution en un temps croissant de manière polynomiale avec la taille du graphe.

Néanmoins, il y a de nombreux algorithmes capables de trouver d'assez bonnes approximations de la modularité maximale dans un temps raisonnable, Fortunato [Fortunato 2010], Leskovec, Lang et Mahoney [Leskovec et al. 2010] en donnent une bonne couverture.

1.2.1.3 Limite de la maximisation de la modularité

En plus de la limite computationnelle citée précédemment, l'optimisation de la modularité a deux autres limites liées à la nature même de la modularité :

D'une part, la modularité peut être optimiste et ainsi surclasser des partitions, car elle repose sur la densité espérée dans une partie d'un graphe aléatoire qui ne contient pas de communautés (voir 1.2.1.1). Or, Reichardt et Bornholdt [Reichardt et Bornholdt 2006] ont montré au travers d'expérimentations numériques que même un graphe purement aléatoire peut contenir une structure de communautés et être partitionné, ce qui augmente la densité espérée. En effet, quand le graphe aléatoire (modèle nul) contient des communautés, le degré total des sommets d_s de ses sous-graphes augmente diminuant donc le second terme de la somme de Q.

D'autre part, la modularité contient une échelle (granularité) intrinsèque qui dépend du nombre total des arêtes du graphe et du degré d'interconnectivité des sous-graphes (clusters de partition). Ainsi, les algorithmes basés sur la maximisation de modularité conduisent souvent au problème de la limite de résolution [Fortunato et Marc 2007], c'est-à-dire la fusion systématique de petits clusters (qui sont réellement des communautés) en plus grands clusters même si ces clusters sont bien définis et peu connectés entre eux, car leur taille est plus petite que l'échelle de la modularité. De ce fait, les communautés plus petites qu'une certaine taille peuvent ne pas être détectées.

1.2.1.4 Multirésolution

Malgré les limites de la modularité, l'optimisation de la modularité est encore utilisée grâce à l'affirmation que la limite de résolution peut être levée une version multirésolution de la modularité, comme celle introduite par Reichardt et Bornholdt [Reichardt et Bornholdt 2006] :

$$Q_{\propto}^{M} = \sum_{s=1}^{m} \left[\propto \frac{l_s}{L} - (1-\alpha) \left(\frac{d_s}{2L} \right)^2 \right],$$

Dans cette variation, un paramètre réglable de résolution noté $\propto$, avec $0 \leq \propto \leq 1$, permet de changer la taille des clusters recherchés, de très larges jusqu'à très petits. Ainsi, en l'absence d'information sur la taille des clusters, les techniques de multirésolution doivent explorer toutes les tailles possibles pour assurer la détection éventuelle de toutes les bonnes communautés (communautés de bonne taille).

On peut prendre comme exemple la méthode décrite par Lancichinetti, Fortunato et Kertész [Lancichinetti et al. 2009], qui proposent une méthode capable détecter des communautés qui se chevauchent tout en respectant leur possible hiérarchie, par exploration locale du réseau, recherchant la communauté naturelle pour chaque nœud. Pendant la procédure, les nœuds peuvent être visités plusieurs fois, peu importe s'ils ont été assignés à une communauté ou non. De cette façon, les communautés qui se chevauchent sont naturellement recouvertes. L'exploration de tous les niveaux hiérarchiques est permise grâce à la variation du paramètre de résolution.

Cependant, des réseaux réels sont caractérisés par la coexistence de communautés de tailles très différentes. De ce fait, il n'y a pas de taille de cluster caractéristique et régler un paramètre de résolution ne peut pas aider. En effet, Lancichinetti et Fortunato [Lancichinetti et Fortunato 2011] le démontrent et donc le problème de la limite de résolution reste posé, car la maximisation de la modularité fusionne les petites communautés (voir 1.2.1.3), mais aussi casse les plus grandes communautés pour en former de plus petites, ce qui rend caduc l'avantage de la multirésolution.

1.2.2 Autres techniques

D'autres classes de techniques ont vu le jour afin de parer à ces problèmes et aux problèmes classiques de data mining, qui sont dans ce cas : détecter les communautés aussi précisément que possible et traiter de très grands réseaux en un temps raisonnable.

1.2.2.1 Partitionnement algorithmique

Dans de la plupart des cas, les communautés sont définies algorithmiquement c'est-à-dire qu'elles résultent d'algorithmes sans définition précise *a priori*, espérant qu'elles reflètent l'intuition de ce que signifie pour un ensemble de nœuds (arêtes) d'être une bonne communauté.

Zhang et Moore [Zhang et Moore 2014], traitent le problème en ayant recours à des outils de la physique statistique, en utilisant la modularité comme un Hamiltonien à température finie et un algorithme efficace de propagation de croyances pour obtenir le consensus de plusieurs partitions avec une grande modularité, plutôt que de chercher une partition qui maximise la modularité. Puis, en appliquant la procédure récursivement, diviser les communautés jusqu'à ce qu'aucune sous-communauté ne peut être trouvée, la structure hiérarchique dans de réels réseaux peut être cernée plus efficacement.

Un autre algorithme, appelé ImDS [Jianshe et al. 2015], optimise une modularité fondée sur la densité en cherchant et fusionnant répétitivement des micros communautés denses en super nœuds jusqu'à ce que la fusion soit impossible.

1.2.2.2 Optimisation multiobjectif

Un réseau, dans le monde réel, peut avoir de multiples partitions potentielles. Par conséquent, une approche plus adaptée est de considérer simultanément plusieurs objectifs conflictuels pour découvrir des communautés complexes et complètes.

La plupart des algorithmes de détection multiobjectif de communautés sont basés sur des algorithmes évolutionnaires. Pizzuti [Pizzuti 2012] propose un algorithme génétique multiobjectif nommé MOGA-Net ; Maoguo, Qing, Xiaowei et Lijia, [Maoguo et al. 2014] décrivent un algorithme d'optimisation multiobjectif d'essaim de particules fondé sur la décomposition, appelé MODPSO.

Zhou, Wang, Luo et Zhang [Zhou et al. 2015] donnent un algorithme d'optimisation multiobjectif par recherche locale, nommé MOLS-Net. Dans cet algorithme deux objectifs opposés, la somme des densités des liens entre les communautés et la somme des densités de liens à l'intérieur des communautés, sont optimisés conjointement pour assurer des interconnections clairsemées et des intraconnections denses.

1.2.3 Détection de communautés dynamiques

Les techniques présentées jusqu'ici, ne prennent pas en compte l'évolution du réseau étudié, elles ne s'intéressent qu'à l'image du réseau à un instant donné (un cliché). Cela peut être suffisant pour certains systèmes, mais pour de nombreux autres et en particulier pour les réseaux sociaux les communautés changent et évoluent de façon diverses et variées (voir Figure 1.2.2) très fréquemment voire n'arrêtent pas de changer au fil du temps.

De plus, le réseau lui-même est sujet à des changements. En effet, des nœuds (individus) peuvent apparaître et former de nouveaux liens accroissant ainsi le réseau, et inversement si des nœuds disparaissent.

Pour prendre en compte cette dynamicité, de nouvelles méthodes ont été développées, bien que l'analyse de communautés dynamiques soit à ses débuts, en s'inspirant des méthodes existantes et en les mettant à jour ou en proposant de nouvelles dédiées à ce problème.

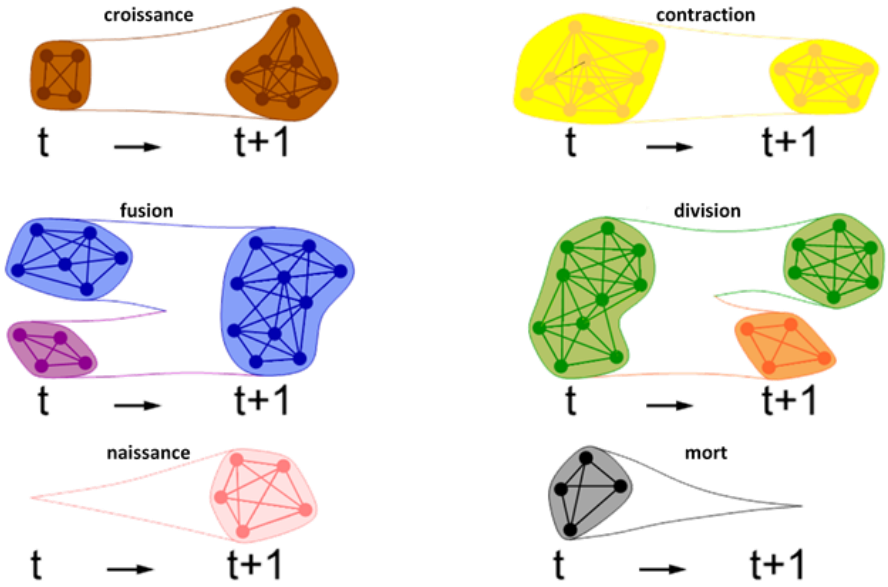


Figure 1.2.2 — évolutions possibles d'une communauté

Une première idée serait d'appliquer une méthode de détection de communautés statiques à chaque cliché du réseau à travers le temps, puis retracer l'évolution des communautés. Mais cela est trop coûteux en temps et en espace.

Étant donné que l'évolution du réseau est souvent progressive, c'est-à-dire qu'il n'y a pas énormément de changement entre deux clichés, il est plus simple de cerner les communautés statiques la première fois puis mettre à jour ce résultat en fonction des modifications du réseau. C'est sur ce principe que s'inspirent la grande partie des algorithmes de détection de communautés dynamiques, la différence entre ces méthodes réside principalement dans la représentation du réseau et de la manière à appliquer les mises à jour.

Dans [Nguyen et al. 2011] et [Li et al. 2012] le réseau est représenté par un simple graphe (un ensemble de sommets et un ensemble d'arêtes) et pour chaque modification d'un élément du graphe (sommet ou arête) un algorithme correspondant à la modification est appliqué pour mettre à jour les communautés existantes. [Nguyen et al. 2011] repose sur le même principe avec l'avantage d'identifier des communautés qui se chevauchent.

Des techniques un peu plus sophistiquées sont disponibles. Par exemple, Schaub, Lambiotte et Barahona [Schaub et al. 2012] utilisent l'évolution temporelle d'un processus de Markov sur le graphe pour révéler des communautés dynamiques pertinentes. Gong, Zhang, Ma et Jiao [Gong et al. 2012] proposent un algorithme immunitaire multiobjectif, qui optimise simultanément la modularité et l'information mutuelle normalisée, ce qui mesure, respectivement, la qualité des partitions et le coût temporel.

Une application possible de la détection de communautés est d'identifier les tendances de contenus générés par des utilisateurs dans un réseau social [Cazabet et al. 2012]. Dans cette application, ce sont les termes présents dans les publications qui constituent un réseau dynamique. Puis, une technique de détection de communautés dynamiques est appliquée sur ce réseau afin d'identifier les tendances et suivre leur évolution.

1.2.4 Simulation à base d'un Système Multi-Agent

Cazabet et Amblard [Cazabet et Amblard 2011] proposent un système multi-agent (SMA) qui rejoue l'évolution du réseau étudié et reproduit l'apparition et la disparition des communautés. Le système est composé :

- d'un environnement qui réifie le réseau et garde la trace de son évolution ;
- d'agents Nœud qui représentent les nœuds du réseau et qui sont capables de créer une communauté, de créer ou effacer un lien avec un autre agent et demander l'intégration de l'agent à une communauté ou son bannissement. Les communautés concernées sont celles auxquelles appartiennent les voisins de l'agent, ce qui représente une petite partie de toutes les communautés ;
- d'agents Communauté représentant les communautés du réseau et décident de l'intégration ou le bannissement d'un agent nœud en se référant uniquement aux agents de la communauté. Ils décident aussi quand il faut intégrer une autre communauté, qui a des nœuds communs, et quand se suicider.

Ainsi, la complexité croît linéairement avec le nombre de liens, et non exponentiellement avec la taille du réseau comme c'est le cas avec les autres techniques, du fait que toutes les opérations sont locales. Ce qui permet le traitement de réseaux de très grande taille.

1.3 Bilan

De très nombreuses techniques de différentes natures sont conçues pour la détection de communautés dans un réseau, et chacune a des spécificités uniques qui les rendent plus adaptées à certaines situations plutôt que d'autres. Donc de tout ce qui a été présenté dans ce chapitre, on peut conclure qu'il n'existe aucune technique universelle qui donne de parfaits résultats en toute situation et en un temps raisonnable. En revanche, plusieurs techniques s'en rapprochent.

Finalement, pour concevoir notre système de diffusion de textos, nous nous inspirons des techniques existantes, et plus particulièrement l'analyse des tendances [Cazabet et al. 2012] et la simulation à base de SMAs [Cazabet et Amblard 2011]. Cela nous permettra de mettre en place en solution capable d'identifier, en un temps raisonnable dans un très grand réseau très dynamique, la communauté d'usagers la plus pertinente pour y diffuser un message donné.

Ainsi donc, le principe de notre technique sera de faire ressortir ou de faire émerger les communautés pertinentes, quelle que soit leur taille, en analysant le contenu des messages envoyés. De plus, il faudra tenir compte de ces messages pour mettre à jour les centres d'intérêt des usagers et suivre l'évolution des communautés. Pour cela, notre système sera basé la théorie des AMAS.

2 Contexte de l'étude

Dans ce chapitre, nous allons présenter les concepts, théories et outils liés aux technologies des systèmes multi-agent (SMA) et des systèmes multi-agent auto-adaptatifs (AMAS) qui seront mis en œuvre pour la réalisation de notre système de diffusion spontané de l'information.

2.1 Systèmes Multi-Agent

2.1.1 Introduction

Un SMA est défini comme un système dans lequel plusieurs entités (agents) autonomes et intelligentes interagissent ensemble dans le but de réaliser un ensemble de buts ou de tâches [Weiss 1995]. Dans ces systèmes, le concepteur délègue une partie du fonctionnement aux entités du système qui ne sont plus des entités fonctionnelles simples, mais des agents. En outre, la conception d'un SMA ne repose pas seulement sur les agents composant le système. Il faut aussi étudier l'environnement dans lequel ils sont immergés, leurs interactions qui permettent le bon fonctionnement du système et la manière dont ils sont organisés pour atteindre leur but.

2.1.2 Concepts associés aux SMA

Dans un premier temps, nous allons définir ce qu'est un agent et ses caractéristiques. Nous étudierons ensuite l'environnement dans lequel ils évoluent ainsi que leurs interactions et leur organisation au sein d'un système.

2.1.2.1 Agent

L'agent est l'élément principal d'un système multi-agent et Ferber [Ferber 1995] a tenté d'en donner une définition minimale. Un agent est une entité physique ou virtuelle *autonome* agissant dans un *environnement* dont il n'a qu'une représentation *partielle.* Une entité est autonome si elle respecte les conditions suivantes :

- Son existence est indépendante de l'existence des autres entités.
- L'entité peut maintenir sa viabilité sans contrôle extérieur dans des environnements dynamiques.
- Son comportement est uniquement fonction de sa structure interne et de ses perceptions.
- Aucun contrôle extérieur à l'entité ne peut avoir accès à sa structure interne.
- L'entité est capable de refuser les requêtes d'autres agents qu'elle perçoit.

Un agent possède :

- *des compétences* : ce qu'il sait faire.
- *des croyances* : ses connaissances sur son environnement, les autres et parfois sur lui-même.
- *des accointances* : ses relations sociales (l'ensemble des agents qu'il côtoie).
- *des aptitudes* : ses capacités de perception, d'action, de décision et d'apprentissage.

Un agent est plongé dans un environnement composé de l'environnement social et l'environnement physique. Le premier est formé des autres agents se trouvant dans le système. Le deuxième constitue l'environnement dans lequel les agents évoluent. Chaque agent suit un cycle de vie de trois étapes [Ferber 1995] :

- *perception* : dans cette étape, l'agent perçoit l'environnement dans lequel il est plongé, et met à jour ses croyances.
- *décision* : en fonction de ses perceptions et de son état actuel, il décide en utilisant ses compétences.
- *action* : suite aux décisions prises, l'agent agit et modifie son environnement pour atteindre son objectif ou s'en rapprocher.

2.1.2.2 Environnement

La notion d'environnement est à considérer selon différents points de vue :

1. Point de vue du SMA : l'environnement correspond à tout ce qui est extérieur au système. Par exemple si l'on considère une fourmilière, l'environnement correspond à tous les éléments de la nature à l'exception des fourmis de la fourmilière.

2. Point de vue de l'agent : l'environnement correspond à tout ce qui est extérieur à lui-même. Par exemple, l'environnement d'une fourmi correspond à l'environnement de la fourmilière et des autres fourmis.

3. Point de vue du concepteur : l'environnement n'a pas de lien direct avec le SMA, il s'agit plutôt de l'environnement d'exécution du programme (système informatique sur lequel s'exécute le SMA), l'environnement de simulation (outils logiciels permettant de simuler, visualiser, évaluer l'exécution du SMA) et l'environnement de développement.

Russel et Norvig [Russel et Norvig 1995] proposent de caractériser l'environnement d'un agent selon les critères suivants :

– *Accessible* : l'agent peut percevoir l'état complet, valide et à jour de son environnement. L'environnement réel et les environnements simulés sont inaccessibles par ce dernier.

– *Déterministe* : l'action d'un agent a un et un seul effet possible sur son environnement. L'état d'un environnement simulé déterministe est donc prévisible en fonction, et seulement en fonction, des actions des agents du système. L'environnement réel est non-déterministe car son état ne dépend pas seulement des actions des agents ; de nombreuses autres variables interviennent (météo, autres agents, etc.).

– *Statique* : lorsqu'un agent décide, l'état de son environnement est inchangé. C'est une contrainte très forte car dans un monde réel, c'est-à-dire dans un environnement dynamique, il est fort probable que l'état du monde perçu soit différent une fois un choix d'action effectué.

– *Discret* : le nombre d'actions que les agents peuvent entreprendre est limité.

2.1.2.3 Interactions

L'interaction, dans un SMA, est la mise en relation dynamique, par le biais d'un ensemble d'actions réciproques, de plusieurs agents ou du système multi-agent et son environnement. Les interactions existant entre les agents influencent leur comportement. Pour rendre possible cette interaction, des mécanismes de communication doivent être mis en place. Nous distinguons deux types de communications :

- La communication *indirecte* : passe par l'environnement, ainsi les agents modifient leur environnement physique pour faire passer des informations aux autres.
- La communication *directe* : passe par l'échange de messages qui peuvent être diffusés à tous (broadcast) ou à un seul (dialogue). Pour pouvoir utiliser ce moyen, un agent doit connaître l'adresse de ces destinataires, et des systèmes de communications doivent être mis en place.

Dans un système multi-agent, les agents, au sein d'un même environnement, interagissent et peuvent amener à former des organisations.

2.1.2.4 Organisations

Une organisation est un ensemble d'individus, regroupés au sein d'une structure régulée, ayant un système de communication pour faciliter la circulation de l'information, dans le but de répondre à des besoins et d'atteindre des objectifs déterminés. En général dans les systèmes multi-agent, l'organisation correspond à la structure du système ou à son architecture. Elle est définie par les liens d'échanges d'information et de requêtes entre les agents du système.

2.1.3 Emergence et auto-organisation

Le type de problèmes auquel nous nous intéressons a la particularité d'avoir une solution a priori « inconnue », ceci est dû en partie à l'aspect dynamique et non linéaire des systèmes étudiés. Résoudre ces problèmes est donc difficile. L'auto-organisation permet de faire émerger une solution adéquate, qui en plus soit adaptative. Nous nous intéressons dans cette partie à deux notions étroitement liées que sont l'émergence et l'auto-organisation, indispensables à l'obtention de la fonctionnalité recherchée.

Émergence

Cette notion vient principalement de l'observation de phénomènes naturels. Le terme émergence désigne l'apparition soudaine et imprévisible de phénomènes au sein d'un système complexe. Suivant [Georgé 2003] l'émergence possède les propriétés suivantes :

- Le phénomène est *ostensible*, il s'impose à l'observateur au macro-niveau.
- Le phénomène est *radicalement nouveau*, il ne peut être expliqué par la connaissance du micro-niveau.
- Le phénomène est *cohérent* et *corrélé*, il a une identité propre mais liée aux parties du micro-niveau.
- Le phénomène crée une *dynamique* particulière, il n'est pas prédéfini, il s'auto-crée et s'auto-maintient. Cette notion est très intéressante dans le cadre des systèmes complexes car bien qu'on ne puisse établir de modèle et donc prédire une solution, il est possible de faire émerger cette solution en se concentrant essentiellement sur l'organisation des parties du micro-niveau.

Auto-organisation

Un Système Multi-Agent s'auto-organise lorsqu'il modifie de lui-même son organisation afin de s'adapter à son environnement. L'auto-organisation est donc synonyme de flexibilité du système car celui-ci est bien plus apte aux changements de l'environnement, c'est-à-dire à même de s'adapter aux perturbations. Il est ainsi primordial que les entités du système ne soient pas contrôlées de l'extérieur [Serugendo et al. 2006] (autonomie) et qu'elles communiquent car c'est bien de leurs interactions que naît une nouvelle organisation.

2.2 Théorie des AMAS

Maintenant que nous avons défini les principales caractéristiques et fonctionnalités des SMA, nous allons présenter un cas particulier de SMA utilisé par l'équipe SMAC : la théorie des AMAS [Bernon et al. 2009]. A l'heure actuelle, compte tenu de la puissance de calcul que les ordinateurs sont à même de nous offrir, nous sommes en mesure de traiter des applications complexes. Cette complexité vient notamment du nombre d'entités en interaction, entités autonomes, hétérogènes et évolutives. Par ailleurs la généricité du matériel informatique rend le nombre de connexions possibles entre différents appareils de plus en plus grand. Ainsi nous devons prendre en compte des environnements dynamiques, des systèmes ouverts et hétérogènes. Bien que des théories formelles permettent de raisonner sur l'espace, le temps et la dynamique d'un monde évolutif, il s'avère que dans certaines situations ces spécifications sont impossibles :

- L'environnement du système est dynamique rendant inopérant l'énumération exhaustive des situations que le système rencontrera.
- Le système a un nombre de composants variable dans le temps, il est ouvert et dynamique.
- La tâche à réaliser par le système est trop complexe pour être spécifiée entièrement ce qui rend une conception complète impossible.
- La manière de réaliser cette tâche est difficile voire impossible à appréhender par un concepteur.

L'approche globale descendante traditionnelle fondée sur des modèles du monde est impossible. Il est donc nécessaire de se baser sur un autre axe de recherche. Ce nouvel axe repose sur le théorème de l'adéquation fonctionnelle [Camps 1998] [Glize 2001] et propose d'utiliser le principe de l'auto-organisation par coopération.

2.2.1 Adéquation fonctionnelle des AMAS

Un système est fonctionnellement adéquat s'il effectue « correctement » la tâche qui lui a été assignée. Le terme « correctement » sous-entend que l'observation est faite du point de vue du concepteur qui connaît la finalité de la tâche.

Théorème. *Pour tout système fonctionnellement adéquat, il existe au moins un système à milieu intérieur coopératif qui réalise une fonction équivalente dans le même environnement.*

Ce théorème montre que tout système réalisant une fonction possède un équivalent fonctionnel au sein duquel les agents qui le composent interagissent de manière coopérative. Ainsi, d'un point de vue conceptuel, il faut se concentrer sur le comportement des agents afin qu'ils disposent de moyens d'interaction coopératifs avec leur environnement.

2.2.2 Processus d'apprentissage par auto-organisation

L'auto-organisation permet de faire émerger la fonction globale d'un système. Cette fonctionnalité est donc dépendante de l'organisation des parties de ce système. Plus formellement, la fonction globale f_s d'un système S est obtenue par combinaison des fonctions partielles f_{pi} assurées par les parties P_i de S. Si l'on prend Θ comme opérateur de combinaison des fonctions partielles, on a alors : $f_s = f_{p1} \Theta f_{p2} \Theta \ldots \Theta f_{pn}$. Comme généralement $f_{p1} \Theta f_{p2} \neq f_{p2} \Theta f_{p1}$, le fait de transformer l'organisation, conduit à changer la combinaison des fonctions partielles et donc à modifier la fonction globale *fs*, devenant elle-même un moyen d'adapter le système à l'environnement. Sur la Figure 2.2.1Figure 2.2.1 — Adaptation par auto-organisation, Θ est illustré par les doubles flèches courbées.

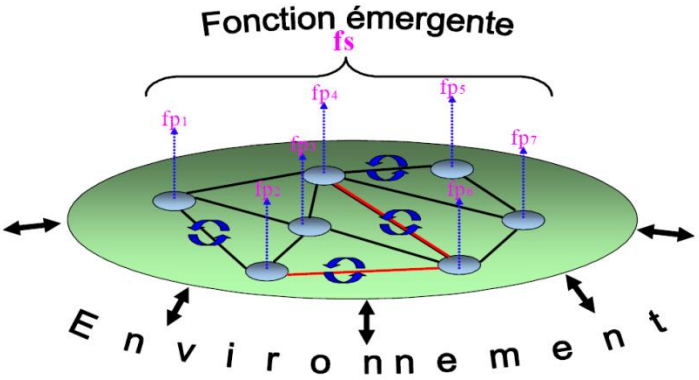


Figure 2.2.1 — Adaptation par auto-organisation

Le processus d'apprentissage par auto-organisation consiste alors à doter chaque partie du système de critères de réorganisation lui permettant de définir elle-même son rôle au sein du système, c'est-à-dire la faculté de modifier de façon autonome ses liens avec les autres parties de façon à faire tendre la globalité du système vers l'adéquation fonctionnelle. Ce phénomène de modification se produit seulement en fonction des perceptions locales que l'agent fait de son environnement. L'auto-organisation permet ainsi d'accroître la robustesse du système, grâce à l'émergence de structures organisationnelles adaptées à des situations non prévues lors de la conception.

2.2.3 Situations Non Coopératives (SNC)

Dans la théorie des AMAS, un agent considère qu'il occupe une place adéquate au sein du système s'il interagit avec autrui coopérativement, ses actions contribuent au processus de résolution et ne nuisent pas à un autre agent; dans le cas contraire, on dit que l'agent est dans une situation non coopérative. Il cherchera alors à atteindre un état qu'il jugera coopératif. Les conditions de non coopération qui conduisent à un processus de réorganisation sont définies de la manière suivante [Georgé 2004] : Non coopération = $\neg C_{per} \vee \neg C_{dec} \vee \neg C_{act}$

- Non coopération de perception ($\neg C_{per}$) telle que :
  - *L'incompréhension* : l'agent perçoit un signal qu'il ne comprend pas.
  - *L'ambiguïté* : l'agent attribue plusieurs interprétations au signal perçu.
- Non coopération de décision ($\neg C_{dec}$) telle que :
  - *L'incompétence* : l'agent n'est pas capable d'exploiter le signal perçu lors de son raisonnement.
  - *L'improductivité* : aucune conclusion n'est produite à partir du signal perçu.
- Non coopération d'action ($\neg C_{act}$) telle que :
  - *La concurrence* : l'agent croit que son action et celle d'un autre vont aboutir au même résultat.
  - *Le conflit* : l'agent croit que son action et celle d'un autre agent sont incompatibles (antinomiques).
  - *L'inutilité* : l'agent croit que son action ne va ni le rapprocher de son but, ni aider un autre agent.

Un agent doit donc détecter les situations non coopératives afin de les traiter et/ou les faire disparaître pour revenir dans un état coopératif. Une étape importante de la théorie AMAS est d'identifier ces SNC selon l'application étudiée, et de définir le comportement des agents face à de telles situations. La méthode ADELFE, présentée ci-après, propose une activité pour faciliter la réalisation de cette étape.

2.3 Méthode ADELFE

Concevoir des logiciels capables de s'adapter à un environnement fortement dynamique impose une méthode de conception rigoureuse qui se distingue de l'approche globale-descendante habituelle. La méthode ADELFE [ADELFE 2000] [Bernon et al. 2002] [Picard 2004] [Gleizes 2004] permet de guider le développeur tout au long de la conception d'un système multi-agent adaptatif conforme à la théorie AMAS.

En se décomposant en cinq phases, ADELFE couvre le processus de développement logiciel dans son intégralité et permet aux concepteurs de revenir aux résultats précédents pour les modifier ou les compléter. Chacune de ces phases se décompose en un ensemble d'*activités (A),* chacune contenant un ensemble de *steps (S).* Comme cette méthode ne concerne que des applications suivant la théorie des AMAS, certaines activités ou étapes ont été ajoutées au RUP.

2.4 Bilan

Actuellement, les applications informatiques deviennent de plus en plus ouvertes et complexes. Les systèmes multi-agent sont une réponse à l'implantation de telles applications. Généralement, de tels systèmes sont plongés dans des environnements dynamiques et il devient nécessaire de les doter de la capacité à faire face aux imprévus provenant de cet environnement. Avec la théorie des AMAS, l'équipe SMAC a cherché à répondre à ces besoins en dotant les systèmes multi-agent de capacité à s'auto-organiser, faisant ainsi émerger leur fonction globale des interactions entre agents les composant au sein de l'environnement.

Toutefois, la conception des systèmes multi-agent, et des AMAS en particulier, pose de nouveaux problèmes aux développeurs. En effet, les agents n'ont pas encore atteint la notoriété des objets et sont moins répandus dans l'industrie. La conception du SMA textOCampus fait l'objet du chapitre suivant.

3 textOCampus

Nous présentons dans ce qui suit le système de diffusion de SMS « textOCampus » développé dans le cadre de ce stage en décrivant la construction de notre système autonome de diffusion de l'information à base d'AMAS étape par étape suivant la méthode ADELFE.

3.1 Besoins préliminaires

Quand un utilisateur autorise textOCampus à diffuser son texto, le système va envoyer le dit texto à tout usager (auditeur) susceptible d'être intéressé par le contenu du message en plus des destinataires de départ que peut expliciter l'expéditeur.

Ces auditeurs sont choisis en fonction des centres d'intérêt communs. Ainsi, chaque message sera diffusé de manière pertinente tout en filtrant les doublons de diffusion et les messages interdits (insultant, raciste, violent…). Pour cela, le système va répartir les utilisateurs dans des groupes d'intérêts communs et chaque auditeur aura la possibilité de retourner un avis sur la pertinence du message qu'il lui est diffusé. Et au vu de l'évolution constante des relations et des centres d'intérêt entre les usagers, le système devra apprendre à cerner ces groupes dynamiques continuellement et en temps réel.

De plus, conformément aux contraintes édictées dans neOCampus [neOCampus 2014], aucun humain ne devrait intervenir pour décider des auditeurs pertinents d'un message, le système devra apprendre en autonomie complète.

3.1.1 Objectifs

3.1.1.1 Objectif général

Développer un AMAS pour l'apprentissage émergent, continu et autonome de l'organisation d'une communauté fortement dynamique.

3.1.1.2 Objectif spécifique au domaine

Proposer un système de diffusion massive et pertinente de l'information en cernant les groupes d'intérêt dynamiques dans le but de faire une synergie de toutes ces informations locales pour améliorer le fonctionnement global de la population des usagers.

3.1.2 Besoins

3.1.2.1 Besoins généraux

Le système multi-agent auto-adaptatif devra :

- Organiser les usagers sous forme de groupes d'intérêt.
- Mettre à jour cette organisation dynamique en fonction :
  - des échanges de SMS.
  - feedbacks ponctuels sur la diffusion.
- Passer à l'échelle.
- être hébergé sur un serveur central et non distribué sur les smartphones des usagers pour des raisons de sécurité, de confidentialité et de consommation en ressources matérielles.

3.1.2.2 Besoins spécifiques au domaine

Diffuser les textos de manière pertinente et efficace (transmettre rapidement la bonne information aux bons usagers sans en omettre un seul) sous plusieurs contraintes :

- Respect de la vie privée et de l'anonymat.
- Pas de surcoût pour les usagers.
- Pas de manipulation supplémentaire par la source.
- Filtrage des messages à caractère insultant, violent, raciste…
- Pas d'ontologie prédéfinie.

3.1.3 Critères nécessaires à l'évaluation

3.1.3.1 Critères généraux

- Robustesse du système : degré de changement de la solution après perturbation (changement dans l'environnement).
- Vitesse d'adaptation du système : temps nécessaire au système pour revenir à état stable suite à une perturbation.
- Complexité temporelle : évolution du temps de traitement nécessaire au système en fonction :
  - du nombre de messages diffusés.
  - du nombre d'utilisateurs.

- Complexité spatiale : évolution de l'espace mémoire utilisé par le système pour l'apprentissage en fonction :
    - du nombre de messages diffusés.
    - du nombre d'utilisateurs.

3.1.3.2 Critères spécifiques au domaine

- Vitesse de convergence de l'apprentissage des groupes d'intérêt.
- Qualité de la diffusion des messages :
    - précision : proportion des messages diffusés à de bons auditeurs parmi tous les auditeurs proposés. Mesure la capacité du système à refuser la diffusion à des auditeurs non-pertinents.
    - rappel : proportion des messages diffusés à de bons auditeurs parmi tous les bons auditeurs qui existent. Mesure la capacité du système à diffuser à tous les auditeurs pertinents.
    - f-mesure : proportion des messages bien diffusés parmi tous les messages diffusés par moyenne harmonique de la précision et du rappel. Mesure la capacité du système à diffuser les messages à tous les auditeurs pertinents et à refuser les autres.

3.1.4 Expérimentations

A l'issue de ce projet il s'agira de décider ou non du passage à une phase opérationnelle sur le campus. Le projet fera donc l'évaluation par des expérimentations sur des scénarios synthétiques de SMS simulant des milliers d'envois de SMS quotidiens entre plusieurs milliers d'usagers disposant de smartphones.

3.1.5 Mots-Clés

- **Texto/SMS :** petit message (160 caractères).
- **Usage/Utilisateur :** personne ayant recours à textOCampus.
- **Source :** expéditeur d'un texto.
- **Auditeur :** utilisateur qui reçoit un texto diffusé par le système.
- **Diffuser un texto :** transmettre le texto à d'autres usagers en plus du ou des destinataires.
- **Communauté :** l'ensemble de tous les usagers.
- **Groupe :** un ensemble d'usagers potentiellement intéressés par les textos de l'un d'entre eux.
- **Feedback/Retour :** avis d'un auditeur sur la pertinence du message qu'il lui est diffusé (intéressé, pas intéressé, message interdit).
- **Message interdit :** message à caractère insultant, violent, raciste, etc.
- **AMAS (Adaptive Multi-Agent System) :** système multi-agent auto-adaptatif.

3.2 Besoins finals

3.2.1 Environnement

L'environnement du système est l'ensemble des utilisateurs se déplaçant dans le monde physique, s'échangeant des textos et pouvant retourner au système des feedbacks ponctuels.

3.2.2 Entités

On peut déterminer 2 types d'entités, une entité active et une entité passive :

- Les utilisateurs : entités actives car ils s'échangent des textos et peuvent, ponctuellement, donner un retour sur la pertinence d'un texto qui leur est diffusé en interagissant avec le système.
- Les textos : entités passives car ce sont de simples messages échangés par les utilisateurs.

3.2.3 Contexte

3.2.3.1 Flots de données entre les entités passives et le système

On peut distinguer un type de flot de données qui doit permettre au système d'obtenir les informations (expéditeur, destinataires, contenu) des textos échangés après autorisation préalable des utilisateurs.

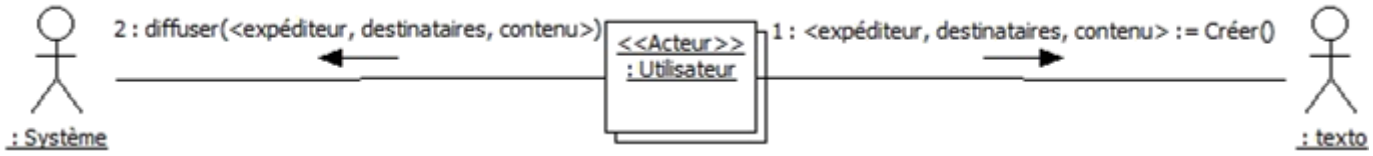


Figure 3.2.1 — Diagramme de collaboration

3.2.3.2 Interactions entre les entités actives et le système

Les utilisateurs sont capables :

- de recevoir des textos ;
- de consulter les textos diffusés par le système ;
- de retourner au système leurs avis (intéressé, pas intéressé, message interdit) sur message diffusé.
- d'envoyer :
  - un nouveau texto à diffuser.
  - une réponse à un texto diffusé.
  - un texto diffusé (transfert).

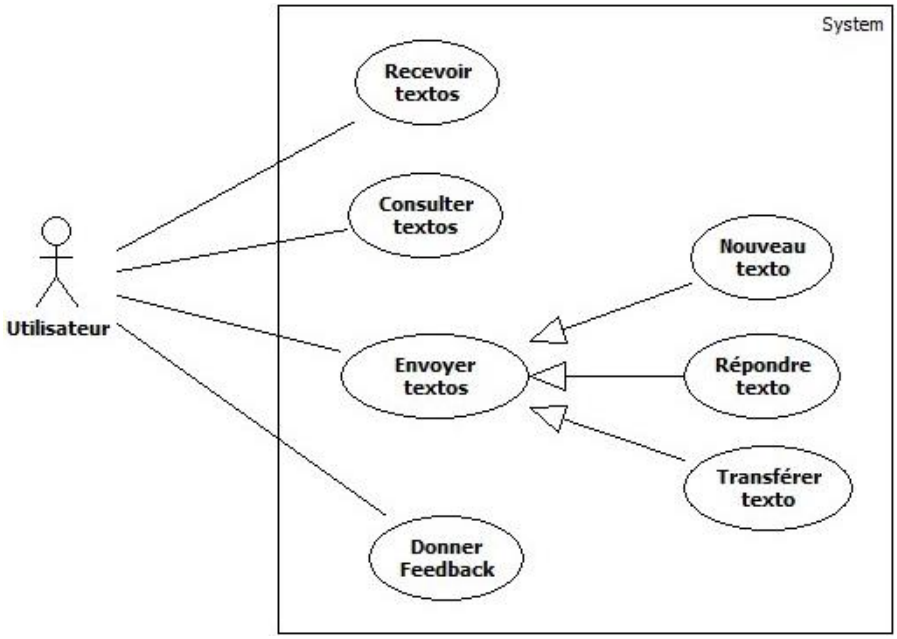


Figure 3.2.2 — Diagramme de cas d'utilisation

3.2.4 Caractérisation de l'environnement

- **Dynamique** : les utilisateurs sont totalement imprévisibles. Ils peuvent envoyer n'importe quel texto à n'importe qui et à n'importe quel moment.
- **Non accessible** : bien que les textos (expéditeur, destinataires, contenu) soient parfaitement connus, les feedbacks occasionnellement disponibles.
- **Non déterministe** : la réaction des auditeurs, aux messages diffusés par le système, est imprévisible. Ils peuvent envoyer, s'ils le désirent, l'un des 3 feedbacks.
- **Discret** : le système peut :
  - recevoir et envoyer (diffuser) des textos de taille limitée (160 caractères max).
  - Recevoir des feedbacks d'utilisateurs (intéressé, pas intéressé, message interdit).

3.3 Analyse

Après les phases d'expression de besoins, l'analyse consiste à construire une compréhension du système afin de le structurer en termes de composants et de repérer la nécessité de l'utilisation de la théorie des AMAS.

3.3.1 Vérifier l'adéquation des AMAS

3.3.1.1 Vérifier l'adéquation au niveau global

Tout d'abord, la tâche globale ne peut pas être complètement spécifiée : en effet, le résultat que nous souhaitons obtenir est connu (connaître pour un message la communauté la plus pertinente i.e. les personnes les plus intéressées par le sujet), mais les moyens et les solutions à mettre en œuvre pour y arriver ne le sont pas forcément. Ensuite, le système est physiquement et fonctionnellement distribué : les entités évoluent en parallèle et chacune possède un rôle bien déterminé.

De plus, l'environnement du système est dynamique : le système devra faire face en temps réel à l'arrivée des nouveaux messages. Finalement, le système doit être ouvert afin de prendre en compte l'ensemble des événements arrivant en cours de fonctionnement (ajout ou suppression d'un utilisateur). Toutes ces raisons nous dirigent vers une modélisation distribuée où plusieurs entités s'auto-organisent afin d'obtenir le meilleur résultat possible. Cette modélisation doit être facilement déployable sur un environnement, justifiant ainsi l'utilisation de la théorie des AMAS au niveau global.

3.3.1.2 Vérifier l'adéquation au niveau local

Cette étape consiste à déterminer si certains des agents identifiés ont besoin eux aussi d'être implémentés en tant qu'AMAS. Dans le contexte de cette étude, les entités n'ont qu'une connaissance limitée de leur environnement. De plus, chaque entité possède un comportement a priori simple, stable et non susceptible d'évoluer. Pour cela, le développement avec les AMAS au niveau local n'est pas nécessaire.

3.3.2 Identifier les agents

Les entités du système qui peuvent être considérées comme des agents sont :

- les utilisateurs (agents Utilisateur) ;
- les termes (agents Terme);
- les messages (agents Message);

3.3.3 Interactions

3.3.3.1 Envoi de messages

1 : à l'émission d'un nouveau message par un utilisateur, l'agent Utilisateur correspondant (l'expéditeur) crée un agent Message en lui fournissant le contenu du message ainsi que la liste des destinataires du message,

2 : une fois l'agent message créé, il envoie le message à tous les destinataires (mettre une copie du message dans la boîte de réception des destinataires),

Puis, pour chaque terme contenu dans le message :

Si le terme n'appartient pas au lexique de l'expéditeur (nouveau terme) il initialise la pertinence du terme (pertinences_termes[terme]=1) puis,

3 : le terme est créé, s'il n'existe pas, en lui fournissant son rang moyen initial dans le lexique (rang_moyen = rang), qui est calculé selon la pertinence du terme comme suit :

a. rang = 1 ;
b. Pour chaque (terme T, sauf le nouveau terme, dans le lexique) :
    i. Si pertinence(T)*(Nombre de tous les utilisateurs / nombre d'utilisateurs dont le lexique contient T) > pertinence(nouveau terme)* (Nombre de tous les utilisateurs / nombre d'utilisateurs dont le lexique contient le nouveau terme), Alors :
        1. rang = rang + 1 ;

4 : sinon, demande l'ajout de l'expéditeur à la liste des utilisateurs de l'agent Terme, en indiquant à son agent Terme le rang du terme pour l'expéditeur afin de mettre à jour le rang moyen du terme. (rang_moyen = rang_moyen+(rang/taille(utilisateurs[]))),

Sinon,

5 : il augmente la pertinence du terme (pertinences_termes[terme]++) et lance la mise à jour du rang moyen du terme, en indiquant à son agent Terme la différence entre l'ancienne et la nouvelle valeur de son rang

(rang_moyen = rang_moyen+(différence_rang/taille(utilisateurs[]))),

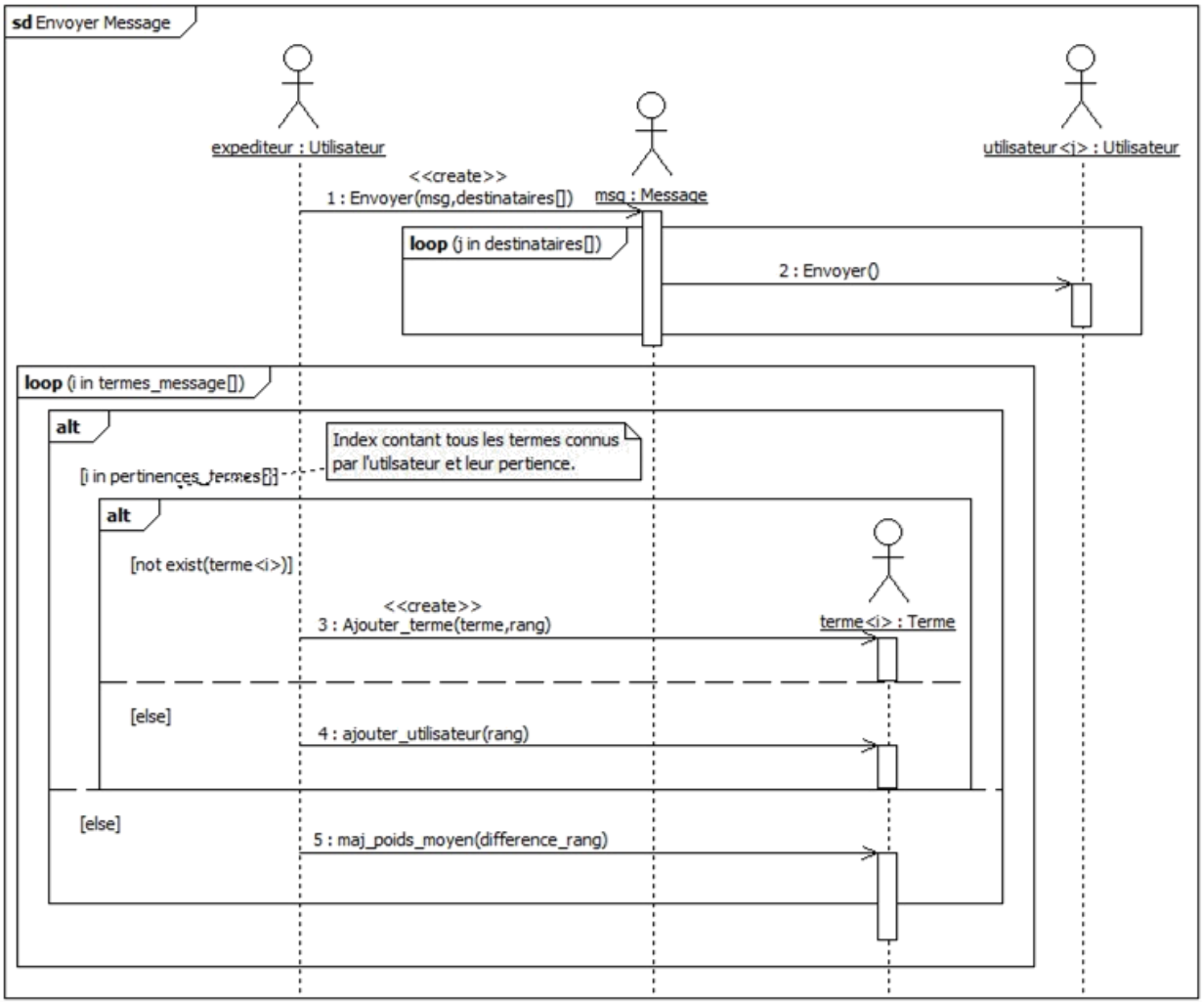


Figure 3.3.1 — Diagramme de séquence pour l'envoi d'un message

3.3.3.2 Diffusion de messages

Après l'envoi d'un message, sa diffusion est lancée si l'utilisateur expéditeur l'a préalablement autorisé.

1 : l'agent Utilisateur expéditeur prévient l'agent Message du début de la diffusion du message en lui envoyant l'index «pertinence_termes[]», qui est une partie de l'index générale de l'expéditeur contenant les termes du message et leur pertinence.

Ensuite, l'agent Message calcule l'écart (désordre) maximale, qui peut être généré pour ce message, comme suit :

a. Si tous les termes du message ont le même rang (premier rang), Alors :
    i. ecart_max = taille du message ;
b. Sinon,
    i. Pour chaque (terme du message) :
        1. Si rang_terme > (rang_max / 2), Alors :
            a. ecart_maxt = rang_max – rang ;
        2. Sinon,
            a. ecart_maxt = rang – 1 ;

2 : ensuite pour chaque terme de l'index, l'agent Message demande à l'agent Terme de lui transmettre sa liste d'utilisateurs et lui passe son rang dans le message, sa pertinence pour l'expéditeur, le rang max et l'écart max.

3 : l'agent Terme retourne à l'agent Message sa liste d'utilisateurs,

4 : puis, l'agent Terme envoie son rang, sa pertinence pour l'expéditeur, le rang max et l'écart max à ses utilisateurs.

5 : pendant ce temps, l'agent Message indique la taille du message, le rang max et l'écart max à tous les nouveaux utilisateurs (non reçus précédemment) contenus dans la liste reçue.

Une fois la taille du message définie, l'agent Utilisateur vérifie si la diffusion du message est terminée et auquel cas prend sa décision d'accepter ou non le message (3.3.3.3 ci-dessous).
De plus, l'agent Message ajoute ces nouveaux utilisateurs à une liste temporaire d'utilisateurs.

6 : si cette liste temporaire n'est pas vide, l'agent Message envoie à chaque agent Terme en relation avec le message, sauf à l'agent Terme qui a envoyé sa liste d'utilisateurs, la liste temporaire des utilisateurs et le rang du terme correspondant dans le message.

Ensuite, l'agent Message ajoute les nouveaux utilisateurs, contenus dans la liste temporaire, à une liste de diffusion générale du message, qui servira à identifier les nouveaux utilisateurs à la réception d'une prochaine liste d'utilisateurs de (3 :).

7 : l'agent envoie son rang dans le message ainsi que son rang moyen à tous les utilisateurs de la liste temporaire, qui mettent à jour leur évaluation du message comme suit :

a. Si rang_moyen > rang_max, Alors :
    i. pertinence_msg = pertinence_msg + |rang_max-rang| ;
b. Sinon,
    i. pertinence_msg = pertinence_msg + |rang_moyen-rang| ;

Puis après chaque mise à jour, l'agent Utilisateur vérifie la fin de la diffusion.

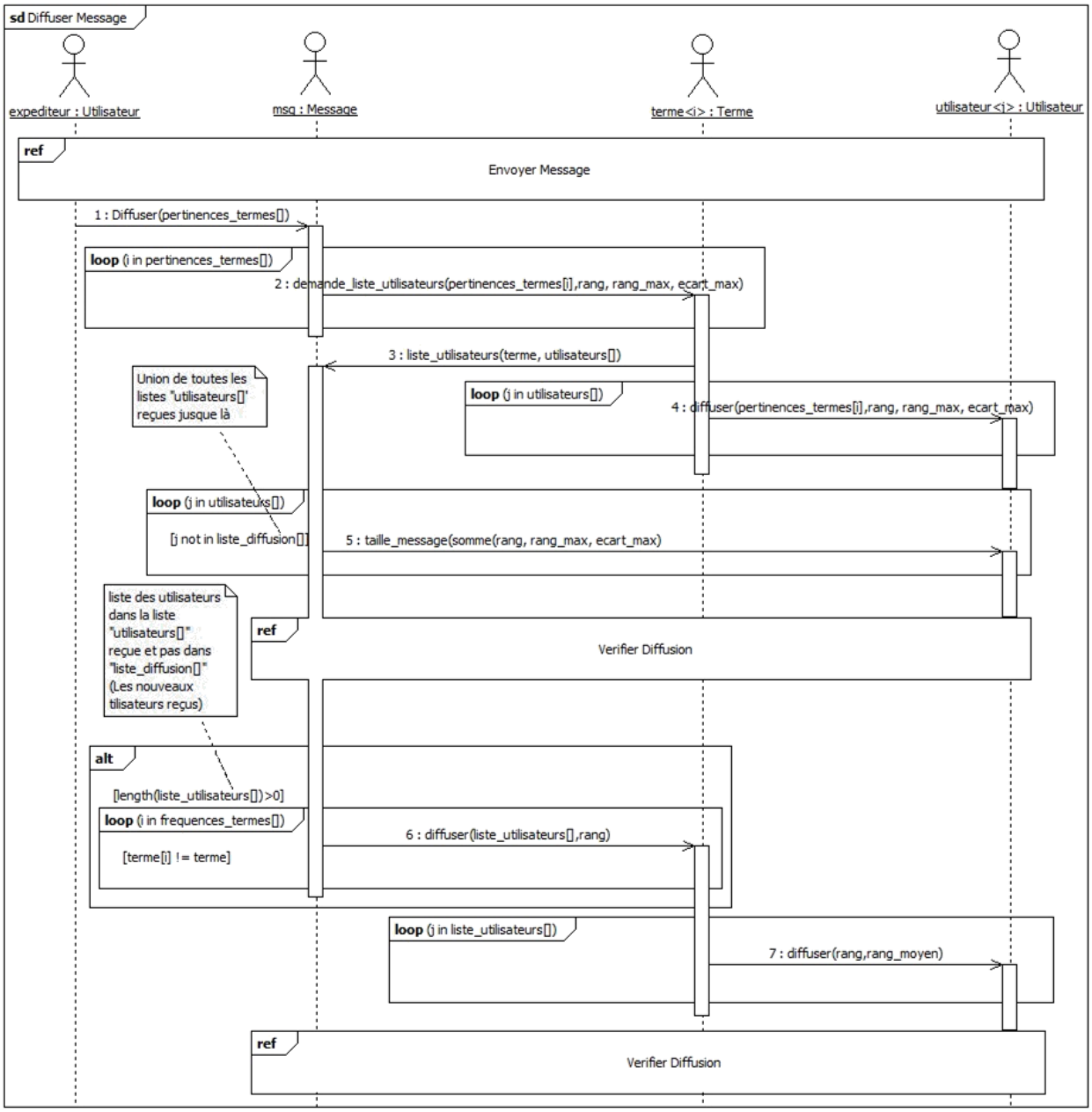


Figure 3.3.2 — Diagramme de séquence pour la diffusion d'un message

3.3.3.3 Vérification de diffusion

1 : Si la pertinence du message est supérieure au seuil d'acceptation (un pourcentage de l'écart max). Alors, l'utilisateur refuse le message.

Sinon, si l'utilisateur a reçu tous les termes du message, alors il calcule et comptabilise les écarts de rang des termes reçus comme suit :

a. Pour chaque (terme reçu) :
   ii. Calculer le rang du terme ;
   iii. pertinence_msg = pertinence_msg + |rang –rang dans le message| ;
   iv. Si pertinence_msg > seuil, Alors :
      1. Refuser = vrai ;

2 : Si le message est refusé, l'agent Utilisateur indique son refus à l'agent Message,

3 : Sinon, l'agent Utilisateur accepte le message et l'indique à l'agent Message correspondant,

4 : qui l'ajoute à la liste de diffusion finale et envoie le message en masquant l'expéditeur.

Finalement, quand l'agent Message reçoit une liste d'utilisateurs de la part de chaque agent Terme et qu'il a bien reçu la réponse de tous les Agents Utilisateur de la liste de diffusion temporaire,

5 : il s'autodétruit pour clore le processus de diffusion.

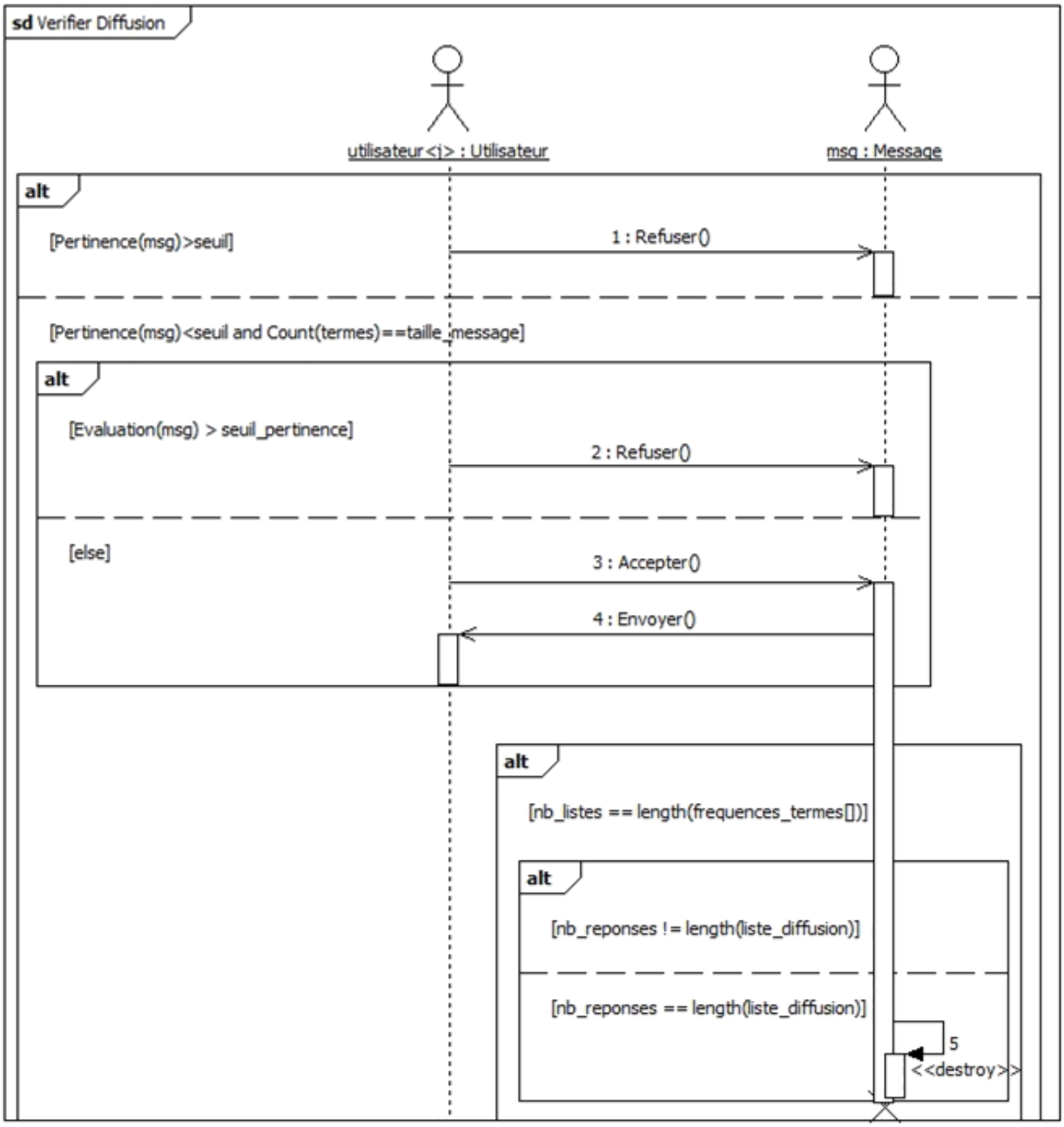


Figure 3.3.3 — Diagramme de séquence pour la vérification de la diffusion d'un message

3.4 Conception

3.4.1 Identifier les agents

L'AMAS se compose de 3 types d'agents :

3.4.1.1 Agent Utilisateur

Désigné par identifiant unique, il représente un utilisateur, peut envoyer un message (créer un agent Message) ou recevoir des messages et décide d'accepter ou non un message diffusé par le système en fonctions du rang des termes qu'il utilise dans ses messages.

3.4.1.2 Agent Terme

Identifié par le terme lui-même, il réifie un terme du lexique et participe à la diffusion des messages dans lesquels il apparaît grâce à la connaissance de :

- la liste des utilisateurs qui l'emploient.
- son rang moyen (moyenne des rangs de tous ses utilisateurs).

3.4.1.3 Agent Message

Identifié par le couple (timestamp de création, identifiant de l'expéditeur), il envoie le contenu du message à tous les destinataires et coordonne sa diffusion, si autorisée, à l'aide de sa connaissance des agents Terme qui composent le message.

3.4.2 Situations Non Coopératives (SNC)

Comme vu précédement (0), les situations non coopératives peuvent conduire les agents à se gêner les uns les autres dans l'accomplissement de leurs tâches respectives et ainsi entraîner à une baisse de performances du système, voire sa complète défaillance.

Dans le but d'éviter c'est cas de figure, les agents doivent adopter une attitude coopérative face à de telles situations. Pour cela, le concepteur doit fournir aux agents des moyens de détecter ces situations, de les résoudre si elles se produisent et au mieux les anticiper.

3.4.2.1 SNC de l'agent Utilisateur

Nom	Conflit
Description	L'agent reçoit le rang moyen d'un terme (Figure 3.3.2 : 7)
Condition	Le rang moyen est supérieur au rang max dans le message
Action	Remplacer le rang moyen par le rang max

Nom	Conflit
Description	L'agent évalue le message
Condition	L'agent accepte le message alors qu'il ne devrait pas
Action	L'agent doit attendre tous les termes du message

Nom	Inutilité
Description	L'agent reçoit un nouveau message de la part d'un autre utilisateur (Figure 3.3.1 : 2) de la part du système (Figure 3.3.3 : 4)
Condition	Le message a déjà été reçu
Action	Ignorer le message

3.4.2.2 SNC de l'agent Terme

Nom	Inutilité
Description	L'agent reçoit une liste de diffusion de la part d'autres agents terme par le biais de l'agent message (Figure 3.3.2 : 6)
Condition	La liste contient des utilisateurs connus du terme. Donc ces utilisateurs ont été traités
Action	Ignorer les utilisateurs connus dans la liste

Nom	Incompétence
Description	L'agent reçoit une liste de diffusion de la part d'autres agents terme par le biais de l'agent message (Figure 3.3.2 : 6)
Condition	La liste contient des utilisateurs inconnus ou créés après le début de la diffusion du message. Donc il ne connaît pas le terme
Action	Envoyer à ces utilisateurs le rang moyen du terme

3.4.2.3 SNC de l'agent Message

Nom	Inutilité
Description	L'agent envoie (Figure 3.3.2 : 6) aux agents termes la liste des utilisateurs reçue (Figure 3.3.2 : 3)
Condition	La liste contient des utilisateurs déjà reçus par d'autres termes (doublons)
Action	Envoyer la liste aux termes, sauf celui qui l'a envoyé, en filtrant les utilisateurs doublons

3.5 Bilan

En suivant les différentes activités de la méthode ADELFE, nous avons réalisé le système textOCampus. Nous pouvons ainsi voir l'adaptabilité de cette méthode pour le développement de systèmes multi-agents adaptatifs. En effet, les différentes activités ajoutées au RUP, nous ont permis de mieux concevoir notre système notamment celles portant sur l'identification des agents, de leurs interactions et de leur comportement.

Concevoir un système en partant des composants et de leurs interactions (démarche bottom up) au lieu de partir de la fonction globale que doit fournir le système (démarche bottom down), est une tâche difficile. Les différentes phases de la méthode ADELFE nous ont aidé à se concentrer sur les différents composants du système nous facilitant ainsi la démarche bottom up utilisée pour la réalisation de notre système.

4 Expérimentations et Analyses

Le système conçu dans le chapitre précédent doit maintenant être évalué afin de valider son utilité. Pour cela, un générateur de scénarios synthétiques de textos simulant des milliers d'envois de textos quotidiens entre plusieurs milliers d'usagers, a été développé et utilisé pour tester notre système et avoir une vue d'ensemble de ses performances.

Dans la première partie de chapitre, nous présenterons notre simulateur. Puis, nous effectuerons une série de tests dont les résultats nous servirons à l'analyse des performances du système de diffusion de l'information.

4.1 Générateur de scénarios

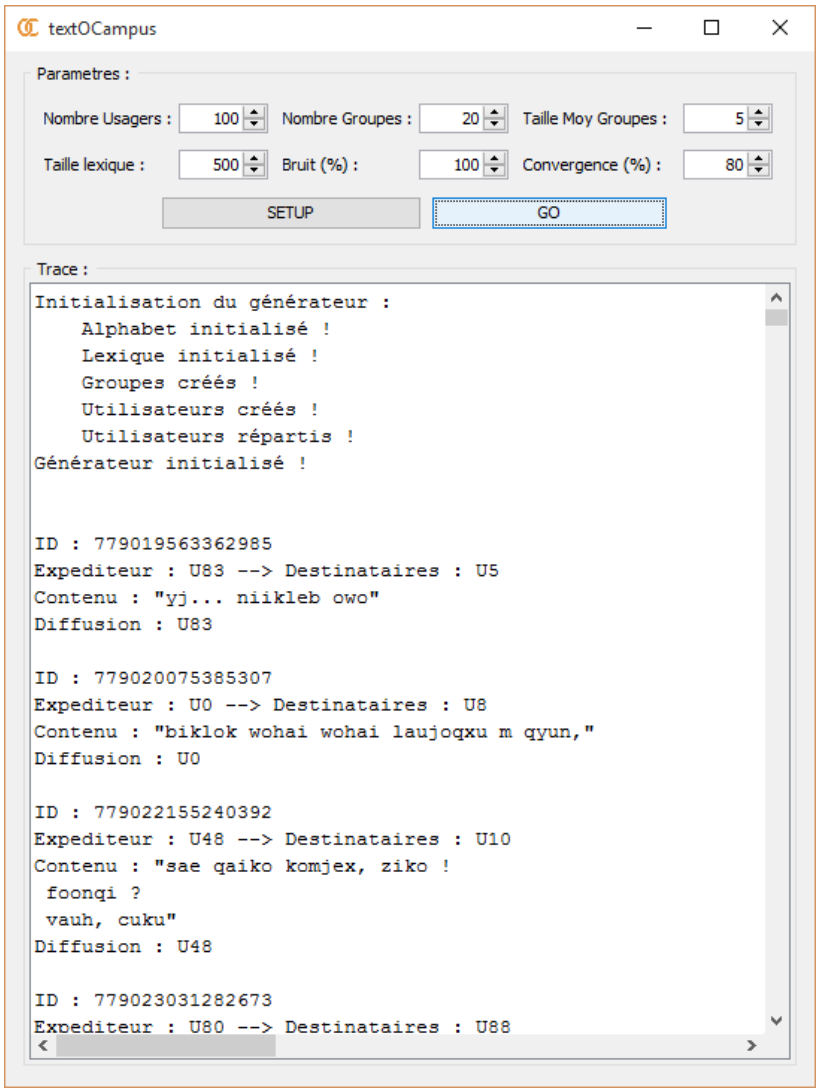
Notre générateur de scénarios d'échanges de textos, nous permet de générer de très grandes populations d'usagers, répartis dans des groupes d'intérêt variés, selon les paramètres de scénarios suivants :

- Le nombre d'usagers ;
- Le nombre de groupes ;
- La taille moyenne des groupes (le nombre moyen d'usagers dans un groupe) ;
- La taille du lexique général ;
- Le pourcentage de couverture : le pourcentage de diffusion à partir duquel on considère que le système converge (apprend à cerner les groupes d'intérêt).

L'initialisation d'un scénario avec les paramètres renseignés, se déroule, suite à la sélection du bouton « SETUP », comme suit :

1) Générer un alphabet de caractères ;
2) Générer un lexique général de la taille de « taille du lexique » ;
3) Créer autant de groupes que le paramètre « Nombre de groupes » l'indique ;
4) Créer des utilisateurs ;
5) Répartir les utilisateurs dans les différents groupes, avec une taille moyenne de « Taille Moyenne Groupe" ;

Une fois le générateur initialisé l'utilisateur peut lancer la simulation d'échanges de textos en cliquant sur le bouton « GO » comme on peut le voir sur la figure suivante :

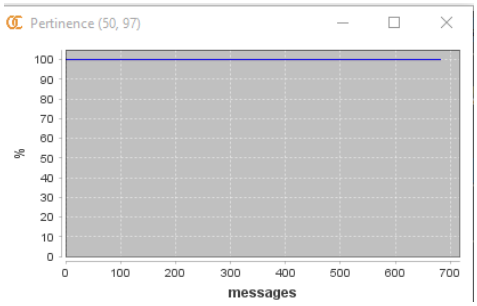


La génération de textos se base sur une chaîne de Markov d'ordre 2 avec une matrice de transition des termes, assignée aléatoirement.

4.2 Expérimentations et Analyses

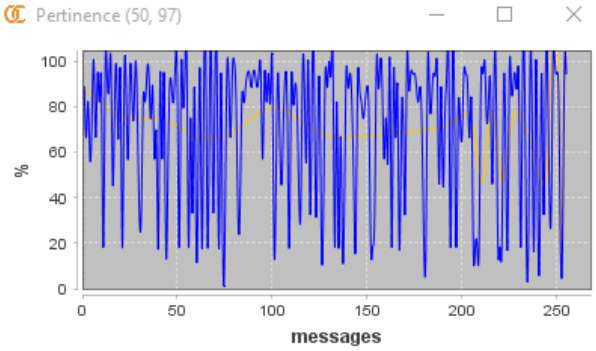
Pour commencer, nous allons étudier l'efficacité de la diffusion des messages. Pour cela, nous générons une population telle que les usagers d'un même groupe possèdent un même lexique i.e. termes identiques avec le même ordre (même rang).

Nous obtenons l'évaluation suivante de la diffusion :



Ce qui correspond à une diffusion parfaite. L'abysse représente le nombre de SMS envoyé dans la simulation et l'ordonnée représente la pertinence de diffusion calculée avec la « f-mesure ». Il est donc compréhensible d'obtenir la couverture de 100%.

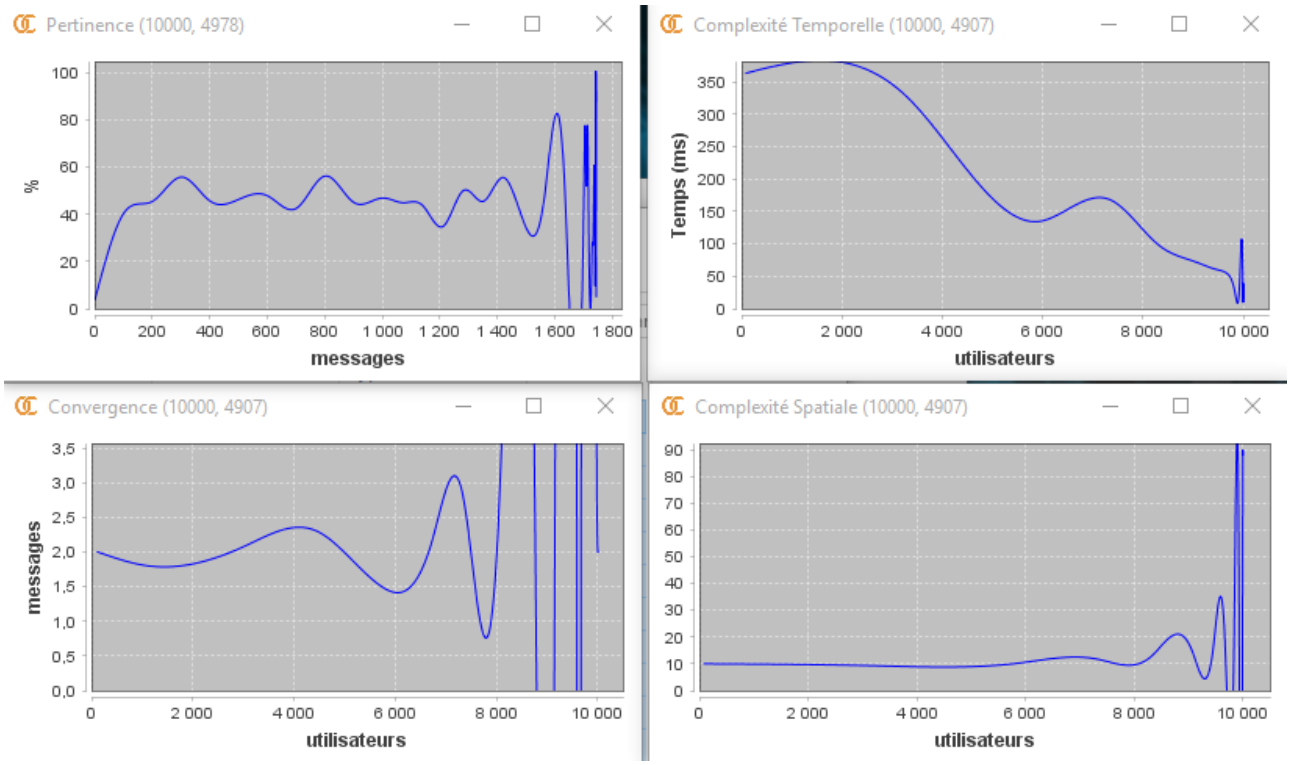
Maintenant nous générons au sein des lexiques des usagers 20% de bruits (désordre des rangs des termes) et nous obtenons le résultat suivant :



On constate une légère dégradation des performances du système. Tel que la courbe bleue représente la fluctuation en temps réel de la performance de la diffusion, alors que la courbe orange représente une moyenne de la satisfaction de cette diffusion sur une fenêtre 20 pas (messages).

Les premières expérimentations se déroulent sur un scénario avec 500 usagers répartis sur 10 groupes de 50 personnes en moyenne et avec un lexique de 1000 termes.

À présent nous allons nous intéresser à l'évaluation des performances d'apprentissage (cerner les groupes) du système sur un scénario avec une population de 10 000 utilisateurs répartis sur 500 groupes de 200 usagers en moyenne et un lexique de 5 000 mots. Nous obtenons les résultats suivants :



Sur les graphiques suivants, les courbes représentent des moyennes sur une fenêtre glissantes de 100 messages.

Le graphique de haut à gauche représente l'évolution de la pertinence de diffusion au fil de l'envoi de messages. On observe une forte croissance au début et à la fin de la simulation et une évolution constante pendant la simulation. Cela résulte du phénomène d'auto-organisation des termes qui va améliorer les performances obtenues.

Le deuxième graphique en haut à droite représente le temps nécessaire à la diffusion des messages. On observe une réduction du temps diffusion, car au début de la simulation les agents sont créés et s'échangent énormément d'informations, ce qui se réduit au fil du temps pour finir pour se stabiliser en un temps de diffusion quasi constant. Il augmente, néanmoins, à l'apparition de nouveaux termes.

Le troisième graphique en bas à gauche, représente le nombre de messages nécessaire afin que le système puisse converger à 40% de la population c'est-à-dire à moins 40% de couverture de groupes d'intérêt.

Le dernier graphique montre la consomption en ressource mémoire pour stocker les informations nécessaires à l'apprentissage du c'est-à-dire la connaissance des lexiques des utilisateurs contentant les termes et leur rang. Cette consommation grandit régulièrement en fonction du nombre des utilisateurs

Remarque :

On observe une forte fluctuation de toutes les courbes et cela à cause de l'arrêt forcé de la simulation qui ne permet à la moyenne d'acquérir assez de données pour former une courbe. D'où l'affichage des valeurs au moment de l'arrêt de la simulation.

5 Conclusion

Les résultats intéressants obtenues lors de stages, ont besoin d'expérimentations plus poussées et sur des cas concrets pour valider les performances de notre système. Mais il est clair pour moi que l'approche multi-agent basées sur la théorie des AMAS permet non seulement de concevoir des systèmes fiables et robustes mais aussi d'obtenir l'émergence de solutions très pertinentes face à des problèmes incomplètement spécifiés.

À l'aide des techniques existantes dans le domaine de la détection de communautés et en suivant la méthodologie ADELFE, nous sommes parvenus à mettre au point un système adaptatif de diffusion de textos à travers de très grands réseaux dynamiques d'usagers. Les propriétés d'adaptation et d'auto-organisation des AMAS, textOCampus peut apprendre en temps réel et en autonomie à cerner les différents groupes d'intérêt dynamiques, qui constituent un réseau.

De plus, grâce aux traitements distribués sur les différents agents du système, la complexité temporelle est relative petite et fixe, de même pour la complexité spatiale qui croît de manière régulière. Cela nous donne un bon indicateur sur la capacité du système à passer l'échelle et à être déployé sur des réseaux du monde réel.

Finalement, nous proposons comme perspective un second mécanisme d'auto-organisation locale à chaque agent utilisateur basé sur l'avis de la diffusion des messages retourné par les utilisateurs vers le système qui va renforcer l'organisation actuelle des termes si le feedback est positif. Et si le feedback est négatif le système devra ajuster l'organisation des termes pour la rapprocher de ce qu'elle devrait être afin de ne plus recevoir de retour négatifs.

Bibliographie

[Georgé et al. 2011] J.-P. Georgé, M.-P. Gleizes et V. Camps, «Cooperation,» chez *Self- organising Software*, Giovanna Springe, 2011, pp. 193-226.

[Newman et Girvan 2004] M. E. J. Newman et M. Girvan, «Finding and evaluating community structure in networks,» *Phys. Rev. E,* vol. 69, n° 2, p. 026113, 2004.

[Chen et Redner 2010] P. Chen et S. Redner, «Community structure of the physical review citation network,» *Journal of Informetrics,* vol. 4, n° 3, p. 278–290, 2010.

[Brandes et al. 2006] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski et D. Wagner, *On Modularity -- NP-Completeness and Beyond,* 2006.

[Fortunato 2010] S. Fortunato, «Community detection in graphs,» *Physics Reports,* vol. 486, n° 3-5, p. 75–174, 2010.

[Leskovec et al. 2010] J. Leskovec, K. J. Lang et M. W. Mahoney, «Empirical Comparison of Algorithms for Network Community Detection,» chez *Proceedings of the 19th International Conference on World Wide Web,* Raleigh, North Carolina, USA, 2010.

[Reichardt et Bornholdt 2006] J. Reichardt et S. Bornholdt, «When are networks truly modular?,» *Physica D: Nonlinear Phenomena,* vol. 224, n° 1-2, pp. 20-26, 2006.

[Fortunato et Marc 2007] S. Fortunato et B. Marc, «Resolution limit in community detection,» *Proceedings of the National Academy of Sciences,* vol. 104, n° 1, pp. 36-41, 2007.

[Lancichinetti et al. 2009] A. Lancichinetti, S. Fortunato et J. Kertész, «Detecting the overlapping and hierarchical community structure in complex networks,» *New Journal of Physics,* vol. 11, n° 3, p. New Journal of Physics, 2009.

[Lancichinetti et Fortunato 2011] A. Lancichinetti et S. Fortunato, «Limits of modularity maximization in community detection,» *Physical Review E,* vol. 84, n° -, p. 066122, 2011.

[Zhang et Moore 2014] P. Zhang et C. Moore, «Scalable detection of statistically significant communities and hierarchies: message-passing for modularity,» *Proceedings of the National Academy of Sciences,* vol. 111, n° 51, pp. 18144--18149, 2014.

[Jianshe et al. 2015] W. Jianshe, H. Yunting, J. Yang, L. Yong, L. Xiaoxiao et J. Licheng, «Density shrinking algorithm for community detection with path based similarity,» *Physica A: Statistical Mechanics and its Applications,* Vols. %1 sur %2218--228, pp. 218--228, 2015.

[Pizzuti 2012] C. Pizzuti, «A multiobjective genetic algorithm to find communities in complex networks,» *Evolutionary Computation, IEEE Transactions on,* vol. 16, n° 3, pp. 418--430, 2012.

[Maoguo et al. 2014] G. Maoguo, C. Qing, C. Xiaowei et M. Lijia, «Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition,» *Evolutionary Computation, IEEE Transactions on,* vol. 18, n° 1, pp. 82-97, 2014.

[Zhou et al. 2015] Y. Zhou, J. Wang, N. Luo et Z. Zhang, «Multiobjective local search for community detection in networks,» *Soft Computing,* pp. 1-10, 2015.

[Nguyen et al. 2011] N. P. Nguyen, T. N. Dinh, Y. Xuan et M. T. Thai, «Adaptive algorithms for detecting community structure in dynamic social networks,» *INFOCOM, 2011 Proceedings IEEE,* pp. 2282--2290, 2011.

[Li et al. 2012] J. Li, L. Huang, T. Bai, Z. Wang et H. Chen, «Cdbia: a dynamic community detection method based on incremental analysis,» *Systems and Informatics (ICSAI), 2012 International Conference on,* pp. 2224--2228, 2012.

[Nguyen et al. 2011] N. P. Nguyen, T. N. Dinh, S. Tokala et M. T. Thai, «Overlapping Communities in Dynamic Networks: Their Detection and Mobile Applications,» *Proceedings of the 17th annual international conference on Mobile computing and networking,* pp. 85-96, 2011.

[Schaub et al. 2012] M. T. Schaub, R. Lambiotte et M. Barahona, «Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation,» *Physical Review E,* vol. 86, n° 2, p. 026112, 2012.

[Gong et al. 2012] M.-G. Gong, L.-J. Zhang, J.-J. Ma et L.-C. Jiao, «Community Detection in Dynamic Social Networks Based on Multiobjective Immune Algorithm,» *Journal of Computer Science and Technology,* vol. 27, n° 3, pp. 455-467, 2012.

[Cazabet et al. 2012] R. Cazabet, H. Takeda, M. Hamasaki et F. Amblard, «Using dynamic community detection to identify trends in user-generated content,» *Social Network Analysis and Mining,* vol. 2, n° 4, pp. 361-371, 2012.

[Cazabet et Amblard 2011] R. Cazabet et F. Amblard, «Using dynamic community detection to identify trends in user-generated content,» chez *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on,* vol. 2, 2011, pp. 402-408.

[Weiss 1995] G. Weiss, «Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence,» *MIT Press,* 1995.

[Ferber 1995] J. Ferber, Les Systèmes Multi-Agent : vers une Intelligence, InterÉditions, 1995.

[Russel et Norvig 1995] S. Russel et P. Norvig, «Artificial Intelligence : a Modern Approach,» *Prentice-Hall,* 1995.

[Georgé 2003] J.-P. Georgé, «L'émergence,» 2003.

[Serugendo et al. 2006] D. M. Serugendo, M.-P. Gleizes et A. Karageorgos, «Self-Organisation and Emergence in Multi-Agent Systems : An Overview,» *Informatica,* n° 30(1), p. 45–54, Janvier 2006.

[Bernon et al. 2009] C. Bernon, D. Capera et J.-P. Mano, «Engineering Self-Modeling Systems : Application to Biology,» I. W. o. E. S. i. t. A. W. (ESAW), Éd., Springer-Verlag, 2009, p. 236–251.

[Camps 1998] V. Camps, «Vers une théorie de l'auto-organisation dans les systèmes multi-agent basée sur la coopération : application à la recherche d'information dans un système d'information répartie,» 1998.

[Glize 2001] P. Glize, «L'adaptation des systèmes à fonctionnalité émergente par auto-organisation coopérative,» 2001.

[Georgé 2004] J.-P. Georgé, «Résolution de problèmes par émergence, Etude d'un Environnement de Programmation émergente,» 2004.

[Picard 2004] G. Picard, «Méthodologie de développement de systèmes multi-agent adaptatifs et conception de logiciels à fonctionnalité émergente,» 2004.

[ADELFE 2000] «ADELFE,» 2000. [En ligne]. Available: http://www.irit.fr/ADELFE/. [Accès le 2015].

[Bernon et al. 2002] C. Bernon, M.-P. Gleizes, G. Picard et P. Glize, «The Adelfe Methodology for an Intranet System Design,» I. B. (. a. C. W. o. A. I. S. (. -. SSE), Éd., Toronto, Ontario, Canada, CEUR Workshop Proceedings, 2002.

[Gleizes 2004] M.-P. Gleizes, «Vers la résolution de problèmes par émergence,» 2004.

[JavAct 2004] «JavAct,» 2004. [En ligne]. Available: http://www.irit.fr/JavAct. [Accès le 2015].

[neOCampus 2014] «neOCampus,» 2014. [En ligne]. Available: http://www.irit.fr/neocampus/. [Accès le 2015].

[Bassett et al. 2013] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson et P. J. Mucha, «Robust detection of dynamic community structure in networks,» *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 23, n° 1, p. 013142, 2013.