HAL
open science

Intégration de l' émergence au sein des systèmes multi-agent Une étude appliquée à la recherche heuristique

Joris Deguet

To cite this version:

Joris Deguet. Intégration de l' émergence au sein des systèmes multi-agent Une étude appliquée à la recherche heuristique. Autre [cs.OH]. Université Joseph-Fourier - Grenoble I, 2008. Français. NNT : . tel-00293838

HAL Id: tel-00293838

https://theses.hal.science/tel-00293838v1

Submitted on 7 Jul 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Joseph Fourier
| _ | _ | _ | _ | _ | _ | _ | _ | _ | _ |

THESE

pour obtenir le grade de
DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

Spécialité : Informatique

préparée au laboratoire LIG – UMR 5217

dans le cadre de l'**Ecole Doctorale EDMSTII**
(École Doctorale de Mathématiques, Sciences et Technologies de l'Information, Informatique)

Présentée et soutenue publiquement
par
Joris Deguet
le 30 mai 2008

Intégration de l'émergence au sein des systèmes multi-agent
Une étude appliquée à la recherche heuristique

Directeurs : Yves Demazeau et Laurent Magnin

JURY

Valérie Camps	Examinateur
Vincent Chevrier	Rapporteur
Yves Demazeau	Examinateur
Jean-Claude Fernandez	Président
Salima Hassas	Examinateur
Dominique Luzeaux	Examinateur
Laurent Magnin	Examinateur
Philippe Mathieu	Rapporteur

Remerciements Je remercie Philippe Mathieu et Vincent Chevrier d'avoir accepté d'être les rapporteurs de mon manuscrit de thèse. Je remercie également les examinateurs Valérie Camps, Salima Hassas et Dominique Luzeaux d'avoir accepté de participer à mon jury. Finalement, je remercie Jean-Claude Fernandez d'avoir accepté de présider à ma soutenance.

Je remercie mes deux directeurs de thèse pour leur encadrement :

- Yves Demazeau pour m'avoir laissé la liberté d'explorer selon ma curiosité ;
- Laurent Magnin pour son accueil à Montréal et pour ses conseils avisés concernant l'implantation.

Je tiens à remercier l'ensemble de l'équipe MAGMA, et dans le détail :

- l'ensemble des doctorants pour avoir partagé avec moi leurs réflexions sur cette situation que nous partagions ;
- ceux qui ont eu la patience d'endurer ma présence dans leur bureau. Dimitri avec qui j'ai découvert la vie de laboratoire et Guillaume qui a plus que comblé le vide laissé par son départ ;
- Xavier pour avoir partagé mes centres d'intérêts, pour s'être acharné à comprendre ce qui nous échappait, pour avoir travaillé avec moi, pour son indéfectible honnêteté qui lui vaut ma confiance ;
- Shadi et Hussein pour ce qu'ils m'ont appris du Moyen-Orient et des subtilités des relations entre Liban et Syrie ;
- Humbert pour avoir toujours sa porte ouverte, pour avoir toujours l'esprit ouvert même s'il s'évertue parfois à convaincre du contraire ;
- Julie pour son sourire de tous les jours et sa disponibilité quand il fallait corriger mon anglais parfois approximatif.

Je remercie mon frère pour ses encouragements fermes à soutenir. Je remercie mon père de ne jamais douter de mes capacités. Je remercie ma mère pour la concision de ses "Travaille mon fils !" mais également pour son français scrupuleux. Merci à Margot et Rose, mes nièces d'être des petites filles joyeuses et vives qui peuvent à tout moment arracher un sourire. Je remercie mes grands mères de ne m'avoir jamais trop reproché mon absence en temps de thèse.

Je remercie Marie-Claude pour avoir pardonné mes oublis dans les tours de vaisselle, pour avoir partagé ma vie pendant la thèse et pour bien vouloir continuer à la partager.

Je dédie ce manuscrit à mes grands pères, Norbert et René qui sont morts pendant ma thèse, qui me manquent.

Résumé L'émergence et les systèmes multi-agent sont deux domaines aux problématiques proches. A travers l'étude de l'émergence dans le cadre des systèmes multi-agent, notre travail consiste à envisager leur principal point commun : la recherche d'un avantage collectif gagné grâce aux interactions entre les agents, quand le résultat global du système qui découle de l'exécution des agents est attribuable à l'interaction. Cette situation est souvent décrite comme celle d'un tout supérieur à la somme de ses parties. Une contribution de cette thèse est d'identifier des interactions dont l'impact peut être évalué à travers la comparaison de systèmes utilisant ou non ces interactions. Nous définissons de tels gains collectifs pour la résolution de problèmes par recherche heuristique dans un modèle d'agents hiérarchique. Ce travail inclut une modélisation multi-agent de ce type de recherche permettant de mettre en évidence des gains collectifs que nous appelons synergies. Trois synergies sont envisagées : la synergie entre des heuristiques travaillant sur le même problème, celle entre des problèmes proches et la synergie entre un utilisateur et le système de recherche artificiel. Ces possibilités de synergies sont à replacer dans le cadre de la discussion concernant l'émergence. Dans ce cadre, certaines populations d'heuristiques correspondent à l'idée de "vrai composite" qui désigne des systèmes composés dont le résultat est attribuable aux interactions entre composants et non à leur composition. L'interaction entre niveaux globaux et locaux s'envisage également naturellement à travers les niveaux induits par l'organisation hiérarchique de notre modèle. C'est en ce sens que le travail fourni contribue à l'étude de l'émergence : par l'étude des possibilités de définition offertes par un modèle multi-agent.

Abstract Emergence and multi-agent systems are two domains with share similar problems. Through the study of emergence in the context of multi-agent systems, this work is centred on their principal common point ; the search for a collective advantage gained through the interaction among the agents, when the global result is due to interaction. This situation is often summarized by a whole that is more than the sum of its parts. One contribution of this thesis is identifying interactions whose impact can be evaluated through the side-by-side comparison of systems including these interactions or not. Such collective benefits are defined for problem-solving by a heuristic search with a hierarchical multi-agent model. This work includes a multi-agent model of that kind of search that identifies collective benefits, which are called synergies. Three kinds of synergies are considered : the synergy between heuristics searching the same problem, the one between similar problems, and the synergy between a human user and the artificial search system. These possible synergies are included in the discussion about emergence. In this framework, some heuristic populations match the idea of a "true composite" that is a composed system whose result is due to interactions between components and not to its composition. The interaction between local and global levels fits in the levels induced by the model's hierarchical organisation. This way, a contribution to the study of emergence is given, by the definition possibilities that a multi-agent model allows for emergence.

Table des matières

1 Introduction **1**
1.1 De l'informatique aux systèmes multi-agent 1
1.2 Emergence 2
1.3 Emergence et systèmes multi-agent 2
1.4 Thèse défendue 3
1.5 Synopsis 4

2 Emergences **7**
2.1 Grands axes 7
2.1.1 Sens commun tiré des dictionnaires 8
2.1.2 Causalité descendante 11
2.1.3 Observation et prédiction 15
2.1.4 La complexité à partir de la simplicité 20
2.1.5 Inscription et Interprétation 24
2.1.6 Emergence et Auto-Organisation 27
2.1.7 Le tout est plus que la somme des parties 29
2.1.8 Approche pragmatique 35
2.1.9 Bilan des grandes lignes 39
2.2 Typologies 40
2.2.1 Typologie de Bedau 40
2.2.2 Typologie de l'encyclopédie de philosophie de Stanford 41
2.2.3 Typologie de Deguet Demazeau Magnin 42
2.3 Oppositions 43
2.4 Tables récapitulatives 43
2.4.1 L'exemple du Jeu de la Vie 44
2.5 Reformulation Agent 47
2.6 Bilan sur l'émergence 49

3 Modèle **51**
3.1 Domaine d'inscription : Recherche heuristique 52
3.1.1 Espace de recherche 52
3.1.2 Générateur 53
3.1.3 Testeur 53
3.1.4 Heuristique 55
3.1.5 Métaheuristiques 56
3.1.6 Bilan 57
3.2 Système multi-agent de recherche heuristique 57
3.2.1 Organisation 57
3.2.2 Interactions 59
3.2.3 Agents 62
3.3 Synergies 67
3.3.1 Synergie d'heuristiques 67
3.3.2 Synergie de problèmes 72
3.3.3 Synergie agents-utilisateur 75
3.4 Situation 77
3.4.1 Recherche heuristique multi-agent 77
3.5 Bilan 78

4 Illustration **81**
4.1 Espace de recherche des Réseaux de neurones 82
4.1.1 Forme 82
4.1.2 Sémantique 83
4.1.3 Utilisation 84
4.1.4 Générateurs 85
4.2 Agents heuristiques 86
4.2.1 Algorithme génétique 87
4.2.2 Recuit simulé 90
4.2.3 Recherche tabou 93
4.2.4 Changement de voisinage 94
4.3 Testeurs 96
4.3.1 Testeur du jeu de backgammon 99
4.3.2 Testeur du partage d'information dans une grille . . . 102
4.4 Implémentation 104
4.5 Synergie d'heuristiques 105
4.5.1 Description de la méthodologie des expériences 105
4.5.2 Backgammon 106
4.5.3 Grille 109
4.6 Synergie de problèmes 112

4.6.1 Backgammon 112
4.6.2 Grille 114
4.7 Synergie Agents-Utilisateur 115
4.7.1 Grille 116
4.7.2 Backgammon 116
4.7.3 Bilan synergie Agents-Utilisateur 117
4.8 Conclusion 117

5 Emergences a posteriori 119
5.1 Retour formel sur notre modèle 119
5.1.1 Relation d'englobant 119
5.1.2 Interactions 120
5.1.3 Dynamique Agent 121
5.1.4 Dynamique gros grain 122
5.1.5 Dynamique grain fin 122
5.2 Emergence et synergies 123
5.2.1 Impact du nombre 123
5.2.2 Exécution parallèle ou concurrente 123
5.2.3 Importance de l'objectif 125
5.3 Influence entre les niveaux 126
5.3.1 Agents et Agents 126
5.3.2 Vrai composite 126
5.3.3 Organisation et Agents 127
5.4 Ce qui émerge 127
5.5 Conclusion 128

6 Retour sur les émergences 129
6.1 Inscription et Interprétation 129
6.2 Emergence et Auto-Organisation 131
6.2.1 Organisation d'un agent 131
6.2.2 Organisation d'une structure d'agents 131
6.3 Surprise et Prédiction 133
6.3.1 Déterminisme 133
6.3.2 Information partielle 133
6.4 Bilan 134

7 Conclusion 135
7.1 Recherche Heuristique 135
7.1.1 Contribution 135
7.1.2 Perspectives 136

7.2 Synergies et Emergence 137
7.2.1 Contribution 137
7.2.2 Perspectives 138
7.3 Modèle multi-agent 138
7.3.1 Contribution 138
7.3.2 Perspectives 139

A Citations **141**

B Glossaire **147**
B.1 Emergence 147
B.2 Modèle multi-agent 148
B.3 Recherche Heuristique 148

C Notations algorithmiques **151**

D Règles du jeu du backgammon **155**

Table des figures

2.1 Liens entre les idées associées à la notion d'émergence et les différents auteurs 45
2.2 Liens entre auteurs et différents exemples. 46

3.1 Un testeur comme la composée d'une fonction de simulation et d'une fonction d'évaluation 54
3.2 Une recherche cherchant à aller plus haut part vers la gauche, ce choix empêche l'heuristique de trouver le sommet : une heuristique doit parfois passer par des points de qualité moindre pour trouver un sommet. 56
3.3 Plusieurs agents de différentes classes et leur organisation. . . 58
3.4 Représentation imbriquée et arbre correspondant. 58
3.5 Un seul agent heuristique 59
3.6 Un agent problème et un agent heuristique fils 59
3.7 Un agent problème et plusieurs agents heuristiques 69
3.8 Nouvelle organisation avec l'introduction d'un agent supervisant la recherche automatique. 73
3.9 Version finale de l'organisation en arbre 76
3.10 Un point est simulé puis le comportement produit comparé à celui fourni en référence. 76
3.11 Un exemple de comportement. 77
3.12 Ensemble des agents et types de messages. 79

4.1 Représentation graphique d'un réseau de neurones 83
4.2 Un réseau de neurones et son interface reliés par les identifiants 84
4.3 La table de jeu du backgammon 100
4.4 Vue habituelle et vue en ligne 100
4.5 L'interface entre configuration et réseau calculant sa note . . 101
4.6 Diffusion d'un gradient par la solution fournie. 104

5.1 Relation d'ordre partiel entre les agents en pointillé. En trait plein la relation "père de". 120
5.2 Une unique recherche, les messages sur la même ligne sont traités par le même agent. 121
5.3 Une unique recherche où les transitions entre mémoires sont étiquetées par les messages. 122
5.4 Une recherche avec deux heuristiques en parallèle 124
5.5 Une recherche avec deux heuristiques en concurrence 124

D.1 Le plateau au début de la partie 156
D.2 Le pion blanc peut sortir en utilisant le déplacement de 1 ou 2. Comme il n'y a pas d'autre coup possible, le joueur doit utiliser le déplacement de 2. 158

Liste des Algorithmes

1 Boucle principale d'agent 62
2 Traitement par un agent Probleme d'un message Rapporter . . 66
3 Traitement par un agent Descente d'un message Initier 67
4 Traitement par un agent Descente d'un message Boucler . . . 68
5 Traitement par un agent Descente d'un message Stopper . . . 69
6 Traitement par un agent Probleme d'un message Partager . . . 71
7 Traitement par un agent Probleme d'un message Mandater . . 74
8 Traitement par un agent Genetic d'un message Boucler 89
9 Traitement par un agent Genetic d'un message Partager 90
10 Traitement par un agent Recuit d'un message Partager 91
11 Traitement par un agent Recuit d'un message Boucler 92
12 Traitement par un agent Tabou d'un message Partager 94
13 Traitement par un agent Tabou d'un message Boucler 95
14 Traitement par un agent Variable d'un message Partager . . . 96
15 Traitement par un agent Exhaustif d'un message Boucler . . . 97
16 Traitement par un agent Direct d'un message Boucler 98

Chapitre 1

Introduction

1.1 De l'informatique aux systèmes multi-agent

L'informatique est une entreprise humaine de grande envergure puisqu'il s'agit de se questionner sur le traitement automatique de l'information. L'algorithmique qui est au coeur de l'informatique consiste à formuler les traitements pour qu'une machine puissent les exécuter, la machine la plus courante étant l'ordinateur. Finalement, l'intelligence artificielle regroupe les tentatives visant l'obtention d'une machine intelligente, qui puissent reproduire ou imiter des traitements sophistiqués dont l'humain est capable.

Ces trois domaines ont historiquement débuté dans le cas centralisé qui correspond à l'existence d'une machine qui suit un unique fil d'exécution. Le cas décentralisé correspond à un ensemble de fils d'exécution qui se déroulent les uns aux cotés des autres. Une image naturelle est alors celle de plusieurs agents qui coexistent ; le cas centralisé correspondant à un seul agent. Les motivations de l'étude du cas décentralisé permettent de mieux saisir pourquoi des chercheurs s'y intéressent :

- la plupart des problèmes que nous rencontrons nécessitent les connaissances de plusieurs personnes pour trouver une solution, ce qui suggère que les techniques de résolution de problèmes par ordinateur doivent permettre une telle coopération pour ne pas se restreindre à quelques problèmes ;
- les systèmes naturels sont souvent constitués de multiples entités qui coexistent, ce qui amène à penser que la simulation par ordinateur se fera plus aisément sur le modèle d'une population d'"agents" ;
- les systèmes artificiels tendent à regrouper de nombreux composants artificiels, ce qui suggère que ces systèmes peuvent être conçus comme

des populations d'agents.

Ces trois domaines que sont la résolution de problème, la simulation et la conception de systèmes artificiels sont les trois grands objectifs des chercheurs en systèmes multi-agent.

Ainsi le domaine du multi-agent correspond à l'étude de l'informatique quand la multitude nous intéresse et que cet intérêt vient du constat que la plupart de ce que nous rencontrons autour de nous s'appréhende naturellement comme une population d'agents.

1.2 Emergence

Le terme d'*émergence* comme de nombreux termes utilisés en science peut avoir un sens différent selon le domaine scientifique où il est utilisé. Ces significations peuvent également être éloignées du sens commun que l'on associe au mot.

Dans la langue de tous les jours, l'émergence correspond à une apparition qui surprend par son importance ou son caractère soudain : un sous-marin émerge s'il fait surface de manière soudaine, un pays émergent bénéficie d'une croissante importante qui augmente son envergure internationale, une technologie est émergente si elle se répand très rapidement tout en changeant de manière importante nos habitudes.

En sciences, l'émergence sert souvent à décrire comment une multitude d'entités prises collectivement forme un tout cohérent. Cette cohérence signifie principalement qu'alors, le tout peut être étudié indépendamment de ses fragments, formant ainsi une nouvelle entité pour laquelle des lois, des propriétés et d'autres éléments d'une théorie peuvent être édictés. C'est dans ce sens que l'émergence peut permettre une progression entre les différents domaines scientifiques en partant de la physique vers la biologie puis la sociologie : les sociétés sont constituées d'individus eux-mêmes constitués d'organes constitués de cellules constituées d'atomes mais toutes ces entités sont étudiés plus ou moins indépendamment, sans prendre en compte l'intégralité des lois connues pour le comportement de leurs constituants.

1.3 Emergence et systèmes multi-agent

Les domaines de l'émergence et des SMA peuvent provoquer une certaine frustration par leur manque de définitions consensuelles[1]. Ce problème n'est pas du à l'absence de définitions mais plutôt à une profusion de possibilités.

[1] Concernant les agents, on peut lire la note à la fin de [Hew06].

L'intuition de cette thèse est de concilier des idées issues de l'étude de l'émergence et des éléments concernant la recherche multi-agent dans l'espoir que ce rapprochement inspire de nouveaux éléments dans les deux domaines dont le point commun est l'idée de multitude. Plus précisément, notre démarche tout au long de cette thèse est de tenter d'importer l'émergence en informatique et dans le domaine des systèmes multi-agent.

Le point commun direct entre émergence et SMA est l'importance du collectif. Cette proximité évidente suggère que les différents travaux sur l'émergence pourraient servir la réflexion sur les SMA à travers la caractérisation du collectif. C'est cette idée qui a motivé le travail que nous présentons dans ce manuscrit.

Le modèle de SMA que nous allons décrire a pour objectif de permettre la description de ce type de système où l'on peut effectivement caractériser l'impact de la collectivité. Il s'agit alors de pouvoir définir des systèmes multi-agent pour lesquels l'impact de la multitude existe et peut être mis en évidence.

1.4 Thèse défendue

La thèse défendue dans ce manuscrit peut se résumer à travers deux affirmations principales :

- L'émergence réside dans l'interaction, dans le caractère collectif d'un système. Il est nécessaire de pouvoir exclure l'impact d'autres facteurs pour pouvoir isoler ce point précis.
- L'émergence concerne des entités composites présentant la caractéristique de ne pas être réductibles à leur simple composition.

Il s'agit donc de chercher à savoir si c'est bien l'interaction qui a permis le résultat. Cette idée correspond à l'intuition affirmant que "le tout est supérieur à la somme des parties". Dans cette thèse nous utiliserons le terme de *synergie* pour décrire cette situation : un gain de performance du système attribuable au travail de plusieurs agents.

Cette idée de synergie, nous l'appliquons au domaine de la recherche heuristique. Dans ce cadre, nous tenterons d'explorer les possibilités de synergies entre plusieurs heuristiques, plusieurs problèmes et finalement entre le système artificiel de recherche et son utilisateur. Ainsi nous cherchons à définir l'émergence pour une large classe de problèmes à travers la mise en évidence du rôle de l'interaction.

Cette description est encore très générale, l'objectif principal de ce manuscrit est de préciser cette première formulation en détaillant la démarche

que nous avons suivie. Ces précisions incluent la définition de notre modèle multi-agent qui nous permet d'envisager de manière uniforme systèmes multi-agent et agent, la description d'un système multi-agent de recherche heuristique et enfin les variations de ce système qui nous permettent d'isoler, de mesurer l'impact collectif.

1.5 Synopsis

Nous donnons ici une vue d'ensemble du contenu de cette thèse.

Emergences Nous donnons une première approche de la notion d'émergence sous la forme d'une étude bibliographique. L'ensemble des travaux existants est présenté sous formes de grands axes correspondant à différentes visions de l'émergence. Nous fournissons également des éléments d'analyse et une ébauche de description multi-agent de l'émergence.

Modèle Nous nous concentrons sur la notion de synergie pour la résolution de problèmes par un système multi-agent. Nous décrivons le domaine de la recherche heuristique centré autour d'un espace de recherche équipé de générateurs dans lequel des heuristiques naviguent à l'aide d'un testeur leur indiquant la qualité d'une solution. Nous décrivons notre modèle multi-agent à travers l'exemple de notre système de recherche heuristique dans lequel nous définissons finalement les synergies. En tout, nous opérons une triple restriction : i) des SMA vers notre modèle multi-agent, ii) de la résolution de problèmes vers la recherche heuristique et iii) de l'émergence vers les synergies.

Illustration Nous décrivons les éléments nécessaires à compléter la description du système afin de pouvoir mener les expériences qui nous permettent d'observer des synergies. Il s'agit principalement de décrire l'espace de recherche des réseaux de neurones, les heuristiques qui l'explorent ainsi que les testeurs modélisant nos problèmes d'illustration. Finalement, nous fournissons les résultats concernant les différentes synergies définies précédemment.

Emergence et synergies Nous revenons sur la thèse défendue à la lumière des éléments présentés dans notre modèle et des expériences et résultats. En particulier, nous explorons les spécificités d'une synergie mais

également la possibilité d'obtenir des vrais composites. Finalement, nous envisageons ce qui peut être qualifié d'émergent dans notre étude.

Emergence et SMA Nous revenons sur certaines possibilités de définitions de l'émergence pour lesquelles nous pensons que notre modèle multi-agent peut être utile. Ces suggestions qui peuvent servir de pistes de recherche pour la caractérisation de l'émergence servent également à envisager les particularités de notre modèle multi-agent.

Chapitre 2

Emergences

Introduction

Ce chapitre propose un panorama sur le travail existant concernant la notion d'émergence. Le nombre de travaux produits depuis l'apparition du terme d'émergence en philosophie interdit la prétention d'être exhaustif. Ce chapitre contient notre compréhension actuelle de la notion d'émergence : nous tentons de retranscrire ici la structure que nous percevons à travers les différents travaux, en nous aidant de nos connaissances en informatique[1].

Les éléments que nous présentons dans ce chapitre servent deux objectifs : fournir un portrait structuré de la notion d'émergence mais également refléter le cheminement qui nous a poussé au reste du travail décrit dans cette thèse.

La suite de ce chapitre se compose ainsi :

- Une présentation structurée par un ensemble de grands axes.
- La description des liens entre ces axes à travers des schémas et typologies.
- Une tentative de formulation de ces idées dans un cadre multi-agent centré sur les idées de variables, de fonctions et d'interactions.

2.1 Grands axes

Il existe une multitude de travaux qu'un exposé à plat rendrait difficile à appréhender. Nous tentons ici de regrouper de nombreux travaux en grands

[1] une approche de la notion peut aussi se faire à travers une collection de citations dans un ordre chronologique, le lecteur peut se référer à l'annexe correspondante page 141

axes. Cette partition des différents travaux reflète notre compréhension, toutefois nous pensons qu'elle permet à la fois de couvrir l'essentiel et de résumer efficacement l'état de l'art.

2.1.1 Sens commun tiré des dictionnaires

Le premier élément que nous exposons est le sens commun associé au terme d'émergence. Nous choisissons de considérer les dictionnaires comme les vecteurs de ce sens commun.

Etymologiquement, la famille de mots (émerger, émergent, émergence) vient du latin :

> emergo, si, sum, ere : sortir de, s'élever, apparaître, se montrer, naître, sortir de la dernière misère, se tirer d'un procès, sortir d'embarras, se montrer, émerger, sortir d'une longue obscurité, de dégager des maux.
> [Gaf95, page 585]

Le sens habituel en français tiré du Petit Larousse Illustré est le suivant :

> 1. sortie d'un liquide, d'un fluide, d'un rayonnement hors d'un milieu. *émergence d'une source.*
> 2. apparition plus ou moins soudaine d'une idée, d'un fait social, politique, économique. *L'émergence de l'idée de tolérance au XVIII siècle.*
>
> [Lar96, page 380]

Le mot anglais *emergency* qui signifie urgence partage la même étymologie que le terme d'émergence. Le sens habituel d'émergence en anglais est tiré de *The New Oxford American Dictionary* :

1. The process of coming into being, or of becoming important or prominent : the emergence of the environmental movement, Japan's emergence as a modern state.
2. The process of coming into view or becoming exposed after being concealed : I misjudged the timing of my emergence.
3. The escape of an insect or other invertebrate from an egg, cocoon, pupal case, etc. : the parasite's eggs hatch synchronously with the emergence of the wasp larvae.
4. An outgrowth from a stem or leaf composed of epidermal and subepidermal tissue, as the prickles on a thistle plant.

[McK05]

On retrouve le même sens général de l'apparition d'une entité (invertébré sortant de son oeuf) ou d'une propriété (modernité de l'état du Japon). On trouve la mention de l'apparition d'un *émergent* qui n'était que caché ce qui contraste avec l'idée d'un émergent n'existant pas avant son apparition.

Le Trésor de la Langue Française Informatisé [TLF] contient une partie dédiée à l'usage du terme d'émergence dans le langage scientifique :

1. Physique : Sortie d'un rayonnement, spécialement d'un faisceau lumineux hors d'un milieu. Point d'émergence. Point par lequel un rayon lumineux sort d'un milieu qu'il a traversé
2. Par analogie

ANATOMIE Emergence d'un nerf, d'une artère. Elle [la veine rénale gauche] croise perpendiculairement la face antérieure de l'aorte, immédiatement au-dessous de l'émergence de l'artère mésentérique supérieure (G. GERARD, Anat. hum., 1912, p. 293). Point d'émergence :

> Que, chez un triton, l'on dévie le bout du nerf sciatique pour l'amener près de la racine de la queue, il pourra se former, au point d'émergence, une queue surnuméraire... J. ROSTAND, La Vie et ses probl., 1939, p. 71.

BOTANIQUE Protubérance se manifestant à la surface d'un végétal et provenant du développement massif de cellules superficielles recouvertes par l'épiderme (GATIN 1924).

GEOLOGIE Apparition des eaux de la nappe phréatique sur un versant ou au fond d'une vallée (GEORGE 1970). Le sommet de l'oxfordien sert de surface d'émergence aux eaux qui ont circulé à travers les calcaires (LAPPARENT, Abr. géol., 1886, p. 280).

Conclusion

Cet exposé des définitions de dictionnaire pointe deux questions intéressantes afin de caractériser l'émergence :

- ce qui émerge pouvait soit être caché soit ne pas exister avant d'émerger
- apparaître ne suffit pas pour émerger, il faut satisfaire un critère supplémentaire, être soudain, important ...

Nous avons naturellement utilisé le terme d'*émergent* pour désigner ce qui émerge. Nous continuerons maintenant à utiliser ce terme en ce sens.

D'un point de vue multi-agent, le sens commun ne permet pas de tirer de conclusions concernant l'émergence. L'émergence ne concerne ici que des phénomènes et ne met pas en jeu l'idée d'un collectif.

2.1.2 Causalité descendante

La possibilité de *causalité descendante* est un point central de la discussion autour de l'émergence dans le domaine de la philosophie. Ce terme recouvre l'idée que ce qui apparaît à un niveau supérieur doit exercer une causalité sur le niveau inférieur pour être qualifié d'émergent.

La vision habituelle s'expose en deux points : tout d'abord l'émergent survient sur un niveau de base, ensuite cet émergent présente un pouvoir causal intrinsèque exercé sur ce niveau de base[2]. La concision de cette description se paye par son manque de précision. On peut envisager séparément les deux questions de la *survenance*[3] et de la causalité descendante.

Nous donnons un exemple introductif : les cellules sont les entités de base, elles forment des organes qui forment le corps de l'individu qui se tue ce qui provoque la mort des cellules. L'idée sous-jacente est alors que :

survenance l'individu survient sur ses organes qui surviennent sur ses cellules

causalité descendante l'individu cause la mort des cellules

Nous exposons ici trois points : le premier est un exposé de la situation de la causalité descendante en philosophie, le deuxième correspond à la notion de causalité descendante en sociologie et finalement une approche dans laquelle la causalité est abandonnée : la nécessité descendante.

Point philosophique

La notion de causalité descendante[4] est très présente dans les définitions d'émergence et en particulier chez les philosophes [O'C94, Kim99, O'C00]. Nous essayons de donner un aperçu ici de ce que peut recouvrir cette notion.

Survenance La première notion à appréhender est celle de survenance. Cette notion a deux définitions distinctes étant donnés deux ensembles de propriétés A et B :

- A survient sur B si deux entités indiscernables relativement aux propriétés de l'ensemble B le sont aussi relativement aux propriétés de A.
- A survient sur B si pour toute propriété A1 de l'ensemble A, si une entité x a la propriété A1, alors il existe une propriété de B1 que possède x et que toutes les entités ayant A1 ont aussi.

[2] "avoir un pouvoir causal" signifie "pouvoir être la cause d'un effet"

[3] le fait de survenir

[4] traduction de "Downward Causation"

Nous retranscrivons la comparaison de Cosma Shalizi [Sha] entre survenance et fonction :

> Let's write lSu for the relationship "higher-level property u supervenes on low-level property l". The defining property of supervenience is that if u and v are not equal, then we cannot have lSu and lSv. So, if lSu and lSv, then u = v. But this establishes a many-one relationship mapping each low-level property to a unique high-level one, which is to say the high-level property is a function of the low-level property. Conversely, it is obvious that functional dependence implies supervenience. Hence, the two concepts are identical.

La survenance semble alors correspondre à toutes les propriétés définies comme des fonctions de propriétés de niveau inférieur.

La plupart du temps, on exige une condition supplémentaire. Dans l'approche la plus fréquente [Kim99, Ste99] on trouve l'exigence que ces propriétés soient imprévisibles et/ou irréductibles aux propriétés de bases sur lesquelles elles surviennent. Nous ne nous engageons pas dans une discussion sur la notion de réduction ici et nous concentrons sur la notion d'imprévisibilité. L'imprévisibilité est à envisager en dehors d'un moyen trivial de prédiction qui vient de la définition même de la survenance : si les propriétés de base sont instanciées alors par définition, la propriété émergente est instanciée.

Causalité propre La causalité descendante correspond à la causalité exercée par le niveau supérieur sur le niveau inférieur. Kim propose le terme plus précis de causalité descendante *réflexive* [Kim99, page 26] pour décrire le cas où l'émergent exerce sa causalité précisément sur la base sur laquelle il survient, formant ainsi une sorte de circularité. O'Connor[O'C94, page 15] exprime cette situation ainsi :

> "novel causal influence". This term is intended to capture a very strong sense in which an emergent's causal influence is irreducible to that of the micro-properties on which it supervenes : it bears its influence in a direct, "downward" fashion, in contrast to the operation of a simple structural macro-property, whose causal influence occurs via the activity of the micro-properties which constitute it.

Cette causalité est propre puisque indépendante de celle des constituants et réflexive car elle s'exerce sur les constituants du système.

Approfondissement de l'approche philosophique Le lecteur intéressé peut se référer à l'article de la Stanford Encyclopedia [Sta] qui fournit un résumé mis à jour régulièrement ainsi que des références. Une introduction plus approfondie se trouve dans [Kim99].

Point sociologique

Dans le domaine de la sociologie et particulièrement dans le domaine de la simulation sociale, on trouve un point de vue différent sur la causalité descendante. Ce point de vue est développé principalement dans les travaux de Sawyer [Saw00] mais on peut également citer Gilbert [Gil02] qui présente une description concise de cette approche.

Dans cette ligne de pensée, une propriété émergente est une propriété globale qui résulte de l'activité des agents qui constituent le système au niveau local. Le cas de la causalité descendante correspond alors au cas où cette propriété globale est perçue par les agents qui changent leur manière d'agir. C'est cette influence d'une propriété globale sur les entités locales qui constitue la causalité descendante.

Nous reprenons l'exemple de Gilbert [Gil02] afin d'incarner cette causalité descendante :

1. Une grille carrée contient des agents de deux couleurs ainsi que des cases vides, une couleur représente 70 pour cent de la population d'agents (vert), les 30 pour cent restant de l'autre couleur (rouge). Chaque agent a un seuil de "tolérance" qui représente la proportion de voisins de couleur différente qu'il peut supporter dans son voisinage avant de devenir "malheureux".

2. La même grille mais les agents malheureux peuvent choisir de se déplacer sur une case inoccupée pour tenter de devenir "heureux", on observe alors la formation de cluster sans que ces agents aient connaissance de cluster, le processus aboutit la plupart du temps à un état stable où tout le monde est heureux : les clusters sont qualifiés d'émergents.

3. On ajoute une fonction qui peut influencer le comportement des agents : le taux de criminalité qui est le nombre de crimes commis dans le passé qui dépend des populations présentes dans le voisinage (une probabilité de commettre un crime est attribuée à chaque couleur de population) ; on permet alors de percevoir cette information pour choisir les déplacements.

Cette progression permet de passer d'une situation statique à la description d'une dynamique puis à la modélisation d'une phénomène collectif qui peut influer sur les décisions des agents. Le travail de Sawyer met en avant la nécessité d'avoir une entité distincte des agents dans laquelle persistent des informations globales, qu'il s'agisse d'informations non-agent au sens du temps (mémoire) ou au sens de l'espace (globalité). Dans l'exemple de Gilbert, le taux de criminalité qui influence les agents constitue une mémoire des événements passés qui ne s'inscrit dans aucun des agents et implique donc une nouvelle entité qui maintient cet historique : l'environnement.

Le point de vue que nous qualifions de sociologique décrit donc une position où des agents participent à une dynamique au même titre qu'une entité qui leur est extérieure. Les agents déterminent l'évolution de cette entité qui à son tour influe sur leurs comportements. C'est cette relation environnement[5]-agents qui permet cette causalité descendante qui suit une direction allant du collectif[6] vers l'individu que Gilbert appelle *immergence.*

Nécessité descendante

Abbott [Abb06] définit l'émergence à travers la notion de *nécessité descendante*[7]. L'idée est d'associer un intérêt à l'émergence qui ne soit pas celui de pouvoirs causaux.

Pour Abbott, l'émergence se définit entre un modèle opérationnel (par exemple les règles de l'automate cellulaire) et un phénomène. Il affirme que l'émergence concerne des *épiphénomènes*. Il définit même les épiphénomènes comme des synonymes d'*émergents*, écartant ainsi la possibilité d'avoir une causalité propre à ces éléments.

Un épiphénomène est appréhendé, conceptualisé, perçu de manière indépendante des entités qui l'engendrent. Cependant il y trouve une implémentation ce qui implique qu'il survient sur ces entités.

Alors l'émergent est qualifié comme un concept qui se conçoit de manière indépendante d'un modèle mais qui trouve une implémentation en termes d'éléments de ce modèle. L'exemple central de Abbott est la Turing équivalence du Jeu de la Vie. Le fait que le Jeu de la Vie soit Turing universel permet d'y appliquer les restrictions connues comme la non-décidabilité du problème d'arrêt.

Dans la proposition d'Abbott, le concept survient bien sur le système

[5] au sens de ce qui existe sans être un agent

[6] non au sens de la population mais au sens de ce qui est partagé par toute la population soit l'environnement

[7] traduction de downward entailment

qui l'implémente mais on ne cherche pas de lien de causalité du concept vers l'implémentation. Il convient alors de trouver un autre intérêt à ce niveau supérieur que celui de déterminer l'exécution du système : cet intérêt est alors l'ensemble des propriétés de l'abstraction qui se répercutent sur l'implémentation.

Conclusion et Position SMA

La question de la causalité descendante est omniprésente dans toute la discussion autour de l'émergence. Le premier point que cette section nous permet de préciser est la différence entre le cas où l'on considère des propriétés et celui où l'on considère des entités. Dans le cas de la discussion philosophique, il s'agit de propriétés de différents niveaux concernant la même entité alors que l'approche des sociologues décrit une situation mettant en jeu une entité globale. Il s'agit de savoir si l'on manipule plusieurs descriptions d'une même entité où un système composé d'entités vivant à plusieurs niveaux.

La possibilité d'avoir à la fois survenance et causalité descendante dépend de leur définitions respectives. Toutefois, nous pouvons tout de suite suggérer que dans le cas d'un SMA, ce sont les agents qui peuvent influer sur le futur des agents, soient les entités du système ce qui semble exclure la possibilité que des propriétés aient un pouvoir causal.

2.1.3 Observation et prédiction

Nous présentons ici un ensemble de travaux qui fonde la notion d'émergence dans le cadre de l'observation et de la tentative de prédiction du comportement d'un système existant. Il s'agit donc de propositions qui s'appliquent lors de l'étude d'un système avec pour objectif un modèle permettant de prédire son comportement.

Cariani

Cariani [Car91] a tenté une première proposition de définition de l'émergence dans un système que l'on veut modéliser et simuler. Cette piste effectue une distinction fondamentale entre *émergence combinatoire* et *émergence créative*, le premier cas correspond à l'apparition d'une nouveauté étant une combinaison de primitives existantes dans le modèle quand le second correspond à l'extension de l'ensemble de ces primitives.

La définition concise de l'émergence pour Cariani [Car91, page 779] est la suivante :

> "The emergence-relative-to-a-model view sees emergence as the deviation of the behavior of a physical system from an observer's model of it"

Cette définition recouvre les deux cas : l'émergence créative[8] correspond au besoin d'enrichir la description du système (le modèle) par l'ajout d'un élément (une variable observable) alors que l'émergence combinatoire[9] correspond à un réarrangement du modèle (réorganisation des variables à travers leurs liens, une nouvelle fonction des variables observables).

On note l'abandon de la connaissance du fonctionnement du système puisque la définition ne met en jeu que les comportements comparés du système et de son modèle.

Bonabeau et Dessalles

Bonabeau et Dessalles [BD97] ont décrit une définition qui tentait de réunir dans le même formalisme les deux notions suivantes :

L'émergence de structures de haut niveau quand on observe un "phénomène" de niveau L_{haut} produit par des constituants et de processus définis et vivants à un niveau L_{bas} et que les propriétés de ce phénomène sont "difficiles" ou impossible à déduire des propriétés des constituants et processus de L_{bas}.

L'émergence relative à un modèle correspond au cas où un système naturel ne peut plus être expliqué par le modèle qu'on en a parce que le comportement du système bifurque de la trajectoire prévue par le modèle : un nouveau modèle est nécessaire (on retrouve la vision de Cariani).

Dans les deux cas, l'observation est une condition nécessaire. Dans le premier cas, il est nécessaire d'observer le phénomène ainsi que les propriétés des constituants avant de se demander s'il est déductible. Dans le second, le comportement doit être observé pour être comparé à la trajectoire prévue par le modèle.

Le travail de Bonabeau et Dessalles consiste alors à donner un cadre dans lequel exprimer les notions d'observateur, d'émergence, de niveaux et de modèles.

Détecteur Le premier élément est la notion de *détecteur*. Un détecteur est défini comme n'importe quel appareil donnant une réponse binaire à ce

[8] Cariani utilise également le terme d'émergence sémantique.
[9] Cariani utilise également le terme d'émergence syntaxique.

qui lui est présenté. Chaque détecteur a un état pouvant être actif ou inactif. L'ensemble des détecteurs est noté D.

Outil Un outil est un élément de structure permettant de calculer des propriétés à partir des détecteurs. L'ensemble des outils est noté T.

Complexité Il s'agit ici d'une complexité descriptive, c'est à dire que l'on souhaite avoir la description de l'état du système qui soit la plus concise possible. Les auteurs utilisent une complexité de description du système S sous la forme d'une complexité relative à l'ensemble d'outils T et des détecteurs permettant de décrire le système D. Cette mesure de complexité est notée $C(S|T, D)$.

L'émergence est alors le cas où sur un pas de temps un nouveau détecteur s'est activé mais que la complexité de description du système a décru contrairement à l'intuition suggérant que l'apparition d'une nouvelle propriété devrait rallonger la taille de la description minimale.

On peut alors reconsidérer les deux cas d'émergence :

L'émergence de structures de haut niveau Le détecteur de niveau supérieur s'active mais rend alors redondante la description des états des détecteurs de niveau inférieur ce qui permet la concision de la description.

L'émergence relative à un modèle S'il manque un outil ou un détecteur, la description de la trajectoire du système nécessite la prise en compte direct de cas particuliers. L'ajout d'un nouveau détecteur ou d'une nouvelle combinaison rend la description a posteriori plus concise.

Darley

Le premier travail mettant en jeu le terme de prédiction a été celui de Vince Darley [Dar94] :

> "A true emergent phenomenon is one for which the optimal means of prediction is simulation."

Cette déclaration se réalise par la définition d'une partition sur les moyens de prédiction du comportement d'un système. Les deux sous-ensembles formant la partition sont les moyens de prédiction par simulation d'une part et les moyens de prédiction par compréhension du phénomène d'autre part.

Darley définit alors deux quantités $s(n)$(simulation) et $u(n)$(understanding) comme les quantités de calcul nécessaires à la simulation et à la compréhension. Nécessaire indique qu'il s'agit des quantités minimales prises sur les deux sous-ensembles que nous venons de décrire. On a alors :

non émergence si $u(n) < s(n)$, soit le cas où la compréhension économise du calcul

émergence sinon, soit le cas où la simulation est parmi les moyens optimaux au sens de la quantité de calcul

L'estimation de la quantité de calcul passe alors par l'utilisation de la *complexité de Kolmogorov* ce qui implique la non-calculabilité des deux quantités $u(n)$ et $s(n)$. Il suggère que les automates cellulaires, étant Turing complets, présentent la propriété d'émergence selon l'argument suivant :

> "by considering the halting problem for any complex system which is capable of universal computation, we know that the best (only) means of prediction in such a situation is to "run the program" i.e. perform the simulation." [Dar94]

Comprendre correspond donc au cas où le résultat d'un programme fourni peut être déterminé sans l'exécuter.

Bedau

Bedau définit ce qu'il appelle *weak emergence* [Bed97] que nous traduisons par émergence faible.

> "There is a system S composed out of "micro-level" parts, the number and identity of these parts might change over time. S has various "macro-level" states (macrostates) and "micro-level" states (microstates). The macrostates are constituted wholly out of its microstates. Further, there is a microdynamic, call it D, which governs the time evolution of S's microstates. D is "local" in the sense it depends on "nearby" microstates of a given part."

Un état macro P de S avec la micro-dynamique D est faiblement émergent si et seulement si P peut être dérivé de D et des conditions externes de S mais uniquement par une simulation.

Les conditions externes sont hors du système et incluent également de possibles sources de non déterminisme qui ne font donc pas partie de D. Dérivable uniquement par dérivation signifie que le calcul nécessaire est proportionnel à l'éloignement dans le futur de qu'on souhaite prédire.

Bedau précise que sa proposition concerne tous les moyens de prédiction :

> for P to be weakly emergent, what matters is that there is a derivation of P from D and S's external conditions and *any* such derivation is a simulation.

Ceci rend alors cette définition totalement équivalente à celle de Darley puisqu'il existe alors deux classes d'appareil de prédiction : les simulations et les autres. Les simulations sont les calculs qui prennent un temps proportionnel à celui nécessaire à l'exécution d'un simulateur de référence.

Shalizi

Shalizi [Sha01] a donné en conclusion de sa thèse une notion d'émergence intéressante. L'intuition qu'il cherche à modéliser est celle d'une régularité dans le système. L'idée de sa proposition est que si un sous-processus du système exhibe une régularité alors la prédiction de ce sous-processus sera plus facile que l'irrégulier qui l'entoure.

Shalizi donne une définition de l'émergence fondée sur ses travaux de thèse qui sont nécessaires à la compréhension du formalisme de son travail, nous donnons ici une description informelle :

1. définir une efficacité de prédiction fondée sur la complexité statistique
2. une dérivation est définie pour incarner le principe de survenance
3. un processus émergent est un processus dérivé qui présente une plus grande efficacité de prédiction que l'original
4. un processus p est dit intrinsèquement émergent s'il existe un autre processus dérivé de p émergent, c'est-à-dire qui présente une meilleure efficacité de prédiction

Il est important de noter ici que cette efficacité de prédiction est du processus dérivé pour le processus dérivé : ce processus présente une efficacité de prédiction accru pour son propre comportement mais pas pour celui de l'ensemble du système.

Conclusion

Le choix de définir l'émergence à travers l'observation du comportement d'un système permet l'étude de n'importe quel système. La contrepartie est que la caractérisation du caractère émergent doit se faire indépendamment du système.

L'objectif poursuivi est la découverte d'un modèle décrivant le fonctionnement du système observé avec les deux objectifs classiques de prédiction et

de simplicité. L'objectif de prédiction, le plus important reflète le fait qu'un modèle n'est intéressant que par sa capacité à prédire le futur du système.

La question de l'imprévisibilité se rapporte aux moyens autorisés pour prédire le comportement : simulation, prédiction ...

Shalizi semble en contradiction avec cette première approche puisque il décrit le cas émergent comme précisément celui où un dérivé existe qui permet un gain au regard de la prédiction. Cette contradiction se résout car l'efficacité de prédiction de Shalizi est un compromis entre la taille du modèle et sa capacité à prédire le futur. Il s'agit donc de prédire plus ou moins bien.

Dans le cadre multi-agent, la question est de savoir si la description complète des agents de leur organisation et/ou l'observation de leur passé permettent de prédire leur comportement futur. Dans l'approche de Shalizi, il s'agirait de trouver une description alternative d'un système multi-agent pour laquelle la perte d'acuité de prédiction serait compensée par le gain de concision.

2.1.4 La complexité à partir de la simplicité

L'idée d'un gain de complexité est résumé par la phrase de Holland [Hol97] :

> "The hallmark of emergence is this sense of much coming from little. This feature also makes emergence a mysterious, almost paradoxical, phenomenon smacking of "get rich quick" schemes."

Nous regroupons dans cette idée deux types d'approches : les approches faisant explicitement mention de la notion de complexité et les approches évolutionnistes. Nous commençons par exposer des notions reliées à cette approche en complexité.

Rasoir d'Occam Ce principe de parcimonie consiste à préférer les descriptions les plus courtes. Un exemple classique consiste à considérer que les mouvements dans le système solaire s'articulent autour du soleil et non autour de la Terre. Ceci rend la description des trajectoires plus simple puisqu'il s'agit alors d'ellipses. L'idée se généralise en considérant que pour l'observation d'un système, c'est son modèle le plus simple qui est le meilleur.

Complexité de Kolmogorov La complexité de Kolmogorov d'un mot s est la longueur du plus court programme qui le produit où un programme est un mot pour une machine Turing universelle (une lambda expression par

exemple). Cette complexité dépend de la machine M choisie ce qui pousse à la noter $K_M(s)$ mais la différence entre $K_{M1}(s)$ et $K_{M2}(s)$ pour deux machines M1 et M2 est bornée par une constante. Le problème principal est l'impossibilité théorique de calculer cette complexité, plus précisément il n'existe pas de procédure qui calcule cette complexité pour l'ensemble des mots.

Approches en IA

L'émergence est présente dans la communauté des chercheurs en intelligence artificielle et plus particulièrement dans la communauté des chercheurs travaillant sur des *heuristiques* évolutionnistes. L'idée est que la complexité du résultat de l'évolution est grande comparée à celle du processus d'évolution.

Programmation génétique et intelligence émergente Angeline a décrit une vision de la programmation génétique comme une technique permettant une intelligence émergente [Ang94]. On peut produire la relecture suivante de l'algorithme génétique habituel : d'une population de candidats en interaction émerge une bonne solution au problème fourni. En particulier, au cours de la recherche d'une fonction plusieurs algorithmes tentent de factoriser une part de la solution dans une fonction partielle. Celle-ci est alors utilisable pour la fonction finale que l'on recherche. L'idée est donc que l'algorithme génétique permet de construire des éléments de plus en plus complexes et donc permet d'accroître la complexité.

Idées proches en Intelligence Artificielle De manière générale, lorsqu'on essaie d'établir un modèle à partir de données, pour deux modèles décrivant aussi bien les données, le plus simple est choisi. Il est donc courant de chercher le modèle le plus simple permettant de représenter la complexité des exemples. Cette idée est largement répandue et correspond à l'envie de décrire une réalité complexe à travers un modèle simple.

Dans le domaine des réseau de neurones, une démarche similaire consiste à considérer que les réseaux les plus simples ont les meilleures capacités de généralisation : c'est la motivation des approches de "minimal brain damage" qui simplifient un réseau de neurones en tentant de ne pas diminuer sa performance [CDS89]. Il s'agit donc de tenter de simplifier le modèle en limitant l'impact sur sa capacité de modèle.

Approches explicites autour de l'émergence

Nous faisons ici mention de deux approches cherchant à partir d'une notion de complexité pour en définir l'émergence : Deguet et Demazeau puis Rabinowitz.

Deguet et Demazeau Nous avons cherché à envisager l'émergence comme un gain de complexité observable entre : i) un premier processus responsable de la production d'un phénomène inscrit dans une trace d'exécution et ii) un second processus chargé de sa détection [DD05]. L'objectif est alors de formaliser ces deux processus et de tenter de montrer dans quelle mesure une telle différence de complexité entre la construction et l'observation peut avoir lieu.

Malheureusement, si cette intuition est intéressante, il est difficile de mettre en place une proposition qui soit applicable à des systèmes réels. La principale cause est l'absence d'une complexité qui soit à la fois applicable à des processus et compatible avec l'idée d'une complexité comme concision dans la description. Pour le cas particulier de la complexité de Kolmogorov, nous avons montré qu'une telle définition serait très sensible à la définition de l'observation et de la construction.

Rabinowitz Rabinowitz [Rab05] a décrit l'émergence dans le cadre des théories habituelles de complexité en informatique théorique à savoir la complexité de Kolmogorov et la complexité en temps à travers la profondeur logique de Bennett [Ben88].

L'idée de base consiste à considérer que tout est sous forme de séquences finies de symboles :

- un programme sous forme d'une séquence de symboles
- ce que ce programme produit

Il note alors P le programme et $E(P)$ sa production. La proposition consiste ensuite à déterminer si on peut trouver des descriptions alternatives permettant de dériver $E(P)$ qui soient distinctes de P et qui ne partagent pas d'information avec P.

Pour définir une mesure permettant de distinguer deux algorithmes, l'auteur utilise la mesure d'information mutuelle entre deux séquences de symboles dérivée de la complexité de Kolmogorov K. L'information mutuelle est alors :

$$I(s:t) = K(s) + K(t) - K(< s, t >) \tag{2.1}$$

avec $< s, t >$ la paire ordonnée des chaînes s et t. Intuitivement si $s = t$, alors

la longueur du programme pour produire $K < s, s >=< s, t >=< t, t >$ est très proche de celle du programme pour produire s.

La proposition repose sur la conjecture que les quantités $I(P : Q)$ et $I(E(P) : E(Q))$ ne sont pas forcément corrélées, autrement dit la similarité entre les productions n'est pas la similarité entre les programmes. L'auteur formule ainsi :

> As an example, consider two completely different algorithms for generating the same string. While E(P) = E(Q) implies that I(E(P) : E(Q)) = K(E(P)), it is possible to have I(P : Q) = 0.

Il considère ensuite les différents Q_i tels que $E(Q_I) = E(P)$ c'est à dire que les deux programmes donnent le même résultat.

Définition Supposons que l'on a une fonction $\tau : N \rightarrow P(N)$ alors τ est un compresseur de sensibilité σ(σ-compresseur) s'il existe un $\sigma \in [0, 1]$ tel que pour tout $Q, R \in \tau(n)$:

1. $E(Q) = E(n) = E(R)$
2. $|Q| \leq |n|$
3. $\tau(Q) = \{Q\}$
4. $Q \neq R \Rightarrow I(Q : R) \leq \sigma * min(K(Q), K(R))$

Les compressés de n sont donc au même niveau programme que n, ils produisent les mêmes effets. Ils sont également plus courts que le programme n. Finalement τ engendre des programmes de faible information mutuelle qui produisent le même résultat que n.

L'émergence est alors définie comme le cas où un programme logiquement profond P, et un σ-compresseur τ tel que $\tau[E(P)] \neq \{E(P)\}$. Alors P est σ-émergent relativement à τ si : pour tout $Q_i \in \tau[E(P)]$, $I(P : Qi) \leq \sigma * K(P)$

Rabinowitz affirme que cette définition correspond au cas où le produit de P est compressible par τ et que les compressés ne partagent pas d'information avec P.

Nous précisons avant de fournir notre contre-argument :

1. on exige que $\tau[E(P)] \neq \{E(P)\}$, ce qui veut dire que $E(P)$ n'est pas incompressible.
2. on exige que pour tout $Q_i \in \tau(E(P))$ on a $I(P, Q_i) < \sigma * K(P)$. Ceci s'ajoute au contraintes dues à l'appartenance de Q_i à $\tau(E(P))$, ainsi pour tout $R \in \tau(E(P))$:
   (a) $E(Q_i) = E(E(P))$, c'est à dire que Q et $E(P)$ sont deux algorithmes ayant les mêmes effets

(b) $|Q_i| \leq |E(P)|$, Q est plus court que $E(P)$

(c) $\tau(Q_i) = \{Q_i\}$, Q_i n'est plus compressible

(d) $Q_i \neq R \Rightarrow I(Q_i : R) \leq \sigma.min(K(Q_i), K(R))$

Or l'auteur indique que $E(P) = E(Q_i)$. Ceci nous incite à penser qu'il y a une erreur dans cette formalisation puisque la seule exigence tirée des définitions est que $E(Q_i) = E(E(P))$. En particulier, cette double exigence impliquerait que $E(P) = E(E(P))$ ce qui signifie que l'émergence n'a lieu que quand le programme se produit lui-même.

L'idée sous-jacente à ce travail reste celle de générateurs de $E(P)$ différents de P tout en étant plus courts que $E(P)$. Toutefois, rien n'indique que ce cas soit possible.

Conclusion et Position SMA

L'approche du gain de complexité peut déboucher sur deux types de démarches que nous avons cherchés à exposer ici. Les approches adressant directement la complexité semblent pouvoir, avec une certaine rigueur, désigner une émergence formelle, laquelle parait cependant difficile à appliquer à un système réel. Dans le cas des approches évolutionnistes, le lien avec des systèmes concrets est direct mais la complexité considérée n'est pas identifiée. Par l'exposé de ces deux points de vue, nous avons cherché à suggérer la difficulté à parler de complexité.

Dans le domaine multi-agent et plus généralement dans celui des systèmes complexes, le terme de complexité recouvre en général une autre réalité. Il est souvent question de l'interaction très grande entre les agents qui rend le tout plus complexe que l'étude d'un agent en particulier. Ainsi la complexité est plus souvent associée à l'importance du collectif qu'aux individus de la population. A l'inverse, les approches utilisant une notion formelle de complexité correspondent à la recherche de la description la plus concise d'un comportement, d'observations dans un langage particulier (machine de Turing, réseaux de neurones ...). L'ensemble de ces approches correspond donc à l'étude d'un comportement complexe à partir d'un modèle simple plus qu'à l'étude d'un collectif complexe à partir d'individus simples. Une définition de l'émergence pour les SMA demanderait alors une définition de complexité qui reflète l'importance du collectif.

2.1.5 Inscription et Interprétation

Une des approches les plus répandues dans le domaine des systèmes multi-agent [tc97] consiste à envisager l'émergence dans le cadre d'interac-

tions produisant des *inscriptions* et des *interprétations*. L'émergent est alors considéré comme le produit de tels systèmes.

Forrest

La première apparition de cette émergence est la description donnée par Stephanie Forrest en préface de [For90] où elle décrit un calcul émergent comme :

- une collection d'agents, chacun suivant des instructions explicites (niveau microscopique) ;
- des interactions entre ces agents (dirigées par les instructions) qui aboutissent à des motifs implicites et globaux au niveau macroscopique : des épiphénomènes ;
- une interprétation naturelle de ces épiphénomènes comme calculs.

Dans cette première définition, on note la présence de l'interprétation du motif comme un calcul. Le terme d'épiphénomènes que nous avions déjà rencontré chez Abbott semble exclure la possibilité d'une causalité de ce motif global sur la collection d'agents. Dans ce cadre, l'interprétation sert en fait à mettre en relief le calcul effectué par le système afin de rendre explicite la fonction calculée par les agents.

MR Jean

La conception d'émergence que nous décrivons ici est parmi les plus présentes dans la communauté francophone des systèmes multi-agent. Cette définition est issue d'un travail collectif et retranscrite dans [tc97] et réapparaît dans les travaux de Muller [Mul03], Quinqueton [Qui00], Webber et Pesty [WP02] ou encore Beurier Simonin et Ferber [BSF03].

Cette conception dans sa description initiale suppose un ensemble d'agents en interaction décrits à l'aide d'un vocabulaire ou d'une théorie. Le système produit alors un phénomène (un processus, un état stable ou un invariant) global. On doit alors ensuite avoir observation de ce phénomène global soit par un observateur soit par le système lui-même. Cette observation ne peut se faire qu'à travers une inscription du phénomène d'une part et une interprétation de cette inscription par l'observateur ou par le système d'autre part dans un second vocabulaire ou une seconde théorie. Si l'observation se fait par un observateur extérieur, on a une émergence faible, si le système lui-même observe ce phénomène, on a une émergence forte.

Dans [tc97] deux réalités sont distinguées selon le sens donné au terme d'inscription selon qu'il s'agit d'une description alternative de la même

réalité ou bien d'une entité distincte de l'ensemble des agents :

> "Un autre point soulevé par cette définition est le problème d'interprétation des inscriptions. Ce qui fournit deux sens possibles au phénomène émergent. Un premier sens faible où les inscriptions se réfèrent fondamentalement à un seul et même ordre de réalité appréhendée selon deux niveaux qu'il est commode, voire indispensable de distinguer. Le problème se résume alors à une autre façon de parler des choses plus simple ou plus compacte."

Nous retrouvons ici la distinction habituelle sur le fait que les systèmes sont soit constitués. soit décrits, à plusieurs niveaux.

Muller

Dans [Mul03], Muller précise que l'*environnement* partagé par les agents joue le rôle essentiel de médium d'inscription du phénomène ce en quoi il écarte la possibilité d'un simple alternative de description. L'interprétation du phénomène par les agents implique alors qu'ils maîtrisent un second vocabulaire permettant de décrire le phénomène global.

Il s'agit alors d'agents dont la composition comprend deux vocabulaires, un correspondant au niveau de l'agent et un autre permettant à l'agent de se représenter un phénomène global. Ces phénomènes globaux peuvent alors influer sur le futur comportement de l'agent.

Ce point de vue est quasiment identique à celui de la causalité descendante en sociologie. En effet, dans les deux cas il est nécessaire d'avoir une entité de niveau global qui enregistre ou inscrit les actions des agents donnant lieu à un phénomène global qui une fois interprété par les agents peut ensuite influer sur leur comportement futur.

Il est intéressant de noter que de nombreux travaux dans le domaine de la simulation multi-agent rentrent dans le cadre de l'inscription d'un phénomène global produit par les actions individuelles des agents dans un environnement partagé. La question devient alors de savoir si ce phénomène rétro-agit sur les agents et finalement si cette rétro-action met en jeu l'interprétation du phénomène.

Conclusion et Position SMA

Nous avons décrit ici les travaux qui ont été mis en place autour des notions d'inscription et d'interprétation. Nous avons vu que la distinction habituelle entre un seul système doté de plusieurs descriptions et plusieurs

entités organisées en niveaux se retrouve ici. L'ensemble de ces travaux repose uniformément sur l'idée d'un phénomène global produit par des agents locaux. Dans ce cadre, l'environnement est l'endroit où le phénomène peut s'inscrire alors que le processus d'interprétation nécessite la perception de cet environnement par les agents.

Les formulations des propositions précédentes décrivent une population d'agents plongés dans un environnement partagé. Beaucoup de travaux issus de la communauté de la simulation multi-agent correspondent à cette idée. La question est alors de savoir si oui ou non le phénomène émergent est interprété par les agents. En effet, si l'influence réciproque entre agents et environnement ne pose pas de problème particulier, la caractérisation d'une "interprétation" n'est pas évidente.

2.1.6 Emergence et Auto-Organisation

Les deux notions d'émergence et d'auto-organisation ont souvent été envisagées dans le même cadre, souvent en considérant que l'auto-organisation était le moyen et l'émergence la finalité. Nous donnons ici quatre points de vue sur cette question.

Camps [Cam98] décrit les liens possibles entre auto-organisation et émergence. L'auto-organisation, vue comme l'activité organisante des agents sans intervention extérieure, produit des comportements et des structures collectives au premier rang desquels l'organisation du système. Ceux-ci se situent au niveau collectif et résultent des actions des agents : ils correspondent bien à l'idée d'émergence comme une entité de niveau supérieur résultante de l'activité des agents. L'auto-organisation est donc le moyen qui amène à la construction d'émergents parmi lesquels l'organisation.

De Wolf et Holvoet ont consacré un article à la question des liens entre émergence et auto-organisation[DH05]. Ils partent de la définition suivante de l'auto-organisation :

> "Self-organisation is a dynamical and adaptive process where systems acquire and maintain structure themselves, without external control."

L'emergence est vue comme l'apparition d'"émergents" radicalement nouveaux produits par les agents constituant le système.

La conclusion est que les deux notions sont indépendantes mais que leur combinaison est une piste de recherche prometteuse.

Shalizi [Sha01] a étudié en profondeur l'idée d'auto-organisation dans le cadre de la mécanique computationnelle initiée par Crutchfield [Cru94]. Dans ce travail, Shalizi propose un cadre formel centré autour de la notion de complexité statistique dans lequel il définit la notion d'auto-organisation comme une augmentation de cette complexité statistique. Il discute également du lien entre émergence et auto-organisation :

> "If we compare this criterion for self-organization with the definition of emergence in chapter 11.2, we see that self-organization increases complexity, while emergence, generally speaking, reduces it, or requires us to use it more effectively for prediction. At first glance, then, self-organization and emergence are incompatible, but this is too hasty. Self-organization is something a process does over time, like being stationary, or having a growing variance. Emergence is, primarily, a relation between two processes, one of which is derived from the other, like "has a smaller entropy rate than". By extension, a process has the property of emergence if any of its derived processes is emergent [...] We can now make sense of the way so many authors have linked self-organization and emergence. When something self-organizes, it becomes more statistically complex, i.e., optimal prediction requires more information. A cognitively-limited observer (such as a human scientist) is therefore motivated to look for a new way of describing the process which has a higher predictive efficiency. That is, the desire to describe things simply makes us look for emergent behavior in self-organizing systems."

Shalizi place l'organisation dans la structure causale d'une seule entité ce qui ne correspond pas à l'idée d'une organisation comme la structure reliant différents agents. Par ailleurs, cette structure causale est découverte par l'observation du comportement de cette unique entité.

Halley et Winkler [HW07] ont proposé un continuum entre émergence simple et émergence complexe à travers le degré d'auto-organisation du système. L'émergence est alors assimilée à des propriétés collectives dont la complexité est identifiée au caractère auto-organisé du système. Les auteurs prennent comme définition de l'auto-organisation une dynamique *far from equilibrium*. L'auto-organisation sert donc à mesurer la complexité de l'émergence.

Conclusion et Position SMA

L'auto-organisation peut être considérée comme un moyen de l'émergence ou l'auto-organisation peut être vue comme le facteur de complexité d'un émergent. La grande variété de définitions des deux notions ne permet pas de conclure sur leur lien indépendamment de l'étude d'un modèle dans lequel les deux trouveraient des définitions précises.

Dans le cas d'une organisation désignant la structure d'une population d'agents, l'auto-organisation nécessite de déterminer quand une influence extérieure a provoqué l'augmentation de l'organisation. Il est nécessaire de définir un modèle multi-agent où ancrer les notions d'organisation, d'autonomie (pour caractériser une auto-organisation) ainsi que l'émergence pour y explorer les liens entre ces notions.

2.1.7 Le tout est plus que la somme des parties

Un des intuitions les plus courantes associées à la définition de l'émergence est que le tout peut être supérieur à la somme de ses parties. Un grand nombre de travaux sur l'émergence ont cherché à donner un sens plus précis à cette intuition.

Searle

Searle [Sea92, page 111] différencie deux niveaux d'émergence. Ces deux niveaux sont principalement distingués par les termes de composition[10] et d'interaction :

> "Suppose we have a system, S, made up of elements a,b,c ... For example, S might be a stone and the elements might be molecules. In general, there will be features of S that are not, or not necessarily, features of a,b,c ... [...] Let us call such features *system features*. [...] Some system features can be deduced or figured out or calculated from the features of a,b,c ... just the way these are composed and arranged [...] But some other system features cannot be figured out just from the composition of the elements and environmental relations ; they have to be explained in terms of the causal interactions among the elements. Let's call these *causally emergent system features*. [...] On these definitions, consciousness is a causally emergent property of systems. [...]

[10] le sens de composition correspond au lien entre un système et ses composants, mais pas au sens de la composée de deux fonctions

> This conception of causal emergence, call it *emergent1*, has to be distinguished from a much more adventurous conception, call it *emergent 2*. A feature is emergent 2 iff F is emergent 1 and F has causal powers that cannot be explained by the causal interactions of a,b,c ...”

Searle distingue :

1. Une caractéristique est dite “du système” si elle ne caractérise aucun élément isolé du système.
2. Une caractéristique du système est émergente 1 si sa réduction nécessite de prendre en compte les interactions et pas seulement les compositions.
3. Une caractéristique du système est émergente 2 si elle est émergente 1 et qu'elle n'est pas non plus réductible en tenant compte des interactions entre les parties.

Corning

Corning [Cor02, page 9] propose de définir l'émergence à partir de la notion de *synergie*. Il s'agit du cas où

> “the combined (cooperative) effects that are produced by two or more particles, elements, parts or organisms – effects that are not otherwise attainable”

On peut alors dire que le tout est *différent* de la somme des parties sans lui être *supérieur*. On retrouve l'idée d'interactions entre les parties.

L'émergence est alors considérée comme le sous-ensemble des effets synergiques qui montre une *nouveauté qualitative*. Il décrit cette situation comme celle où les parties s'adaptent pour constituer un tout, fait de parties différentes.

L'article concernant “synergy” dans le “New Oxford American Dictionary” [McK05] donne la définition suivante :

> “the interaction or cooperation of two or more organizations, substances, or other agents to produce a combined effect greater than the sum of their separate effects”

Un exemple donné par Corning est celui d'une voiture, constituée de toutes ses pièces. Séparées, elles ne font rien, une fois assemblées elles donnent lieu à une synergie, c'est à dire un véhicule roulant. Cet exemple sert à montrer que cette organisation n'a pas à être auto-organisation pour exister puisque l'organisation est ici fixée.

Voyelles

Dans le cadre de la vision VOYELLES des systèmes multi-agent [Dem95], le système est décomposé en 4 composantes : Agents, Environnement, Interactions et Organisations. Dans la description habituelle de ce modèle, trois principes sont construits sur ces composantes.

$$SMA = A + E + I + O \tag{2.2}$$

$$SMA = A \tag{2.3}$$

Le dernier principe décrit le résultat du système en définissant l'émergence comme un élément rendant cette description non linéaire[11].

$$F(SMA) = F(A) + F(E) + F(I) + F(O) + emergence \tag{2.4}$$

Il s'agit ici de l'affirmation de la possibilité d'une supériorité de la fonction du tout sur les fonctions des différentes entités le composant. Toutefois, il ne s'agit que d'une description de haut niveau des systèmes multi-agent qui ne garantit pas qu'un modèle correspondant à la vue en VOYELLES puisse exhiber cette propriété.

AMAS/ADELFE

Une autre proposition que nous classons dans cette partie est celle de la méthodologie ADELFE. Cette méthodologie a pour objectif d'aborder le développement de SMA à fonctionnalité émergente. Le postulat sur lequel se fondent la théorie et la méthodologie est qu'un système fonctionnellement adéquat est un système dans lequel les situations de non coopération sont évitées.

Le coeur de la méthodologie est l'identification des *situations non coopératives*. A partir de la description des agents et de leurs interactions, des classes de situations non coopératives sont identifiées comme les situations génériques identifiées dans [Cam98] auxquelles peuvent s'ajouter des situations spécifiques à l'application. A chacune de ces classes est associé un traitement permettant de restaurer la situation coopérative.

Cette situation correspond à un avantage collectif car la base de la méthodologie est qu'un système coopératif est supérieur à un système qui ne l'est pas. Cette idée est déclinée dans de nombreuses applications sur lesquelles les notions de situations non coopératives sont illustrées.

[11] la somme des fonctions n'est pas la fonction de la somme

Formalisation de Kubik

Kubik [Kub03] a proposé une approche formelle de cette idée de supériorité du collectif sur les individualités. Cette approche est fondée sur la modélisation des agents sous la forme d'ensemble de règles formant une grammaire de tableaux isométriques [DFP95, FF96][12].

L'approche consiste en trois étapes :

1. Les agents sont décrits à l'aide de règles de grammaire.
2. Deux systèmes sont définis dont l'un correspond au tout et l'autre à la somme des parties.
3. Ces deux systèmes engendrent deux langages.

Le cas d'émergence est celui où le langage du tout inclut strictement celui de la somme des parties. Les langages engendrés peuvent être vus comme les mondes accessibles par le système. Nous détaillons l'approche adoptée.

Définitions Soit $V = V_T \cup V_N$ un alphabet constitué de terminaux et non-terminaux. Une grammaire formelle est définie comme un quadruplet :

$$G = (V_N, V_T, S, P)$$

avec S l'axiome (le non-terminal initial) et P l'ensemble des productions. Une production décrit comment réécrire un non-terminal avec une éventuelle condition de contexte.

Kubik définit *un système de grammaire coopératif* :

$$G = (V_N, V_T, S, P_1, \ldots, P_n)$$

avec les productions P_i qui définissent l'agent i.

Contrairement aux grammaires formelles habituelles, les règles de réécriture ne modifient pas une chaîne de symboles mais un tableau à deux dimensions. Une propriété d'isométrie est requise pour les règles afin d'éviter le problème de décider comment étendre le tableau. Nous donnons un exemple pour clarifier ce point.

Considérons la règle de réécriture suivante :

$$\underbrace{\begin{matrix} X & X & X \\ X & X & X \\ : & - &) \end{matrix}}_{\alpha} \rightarrow \underbrace{\begin{matrix} : & - & (\\ : & - &) \\ : & - & (\end{matrix}}_{\beta}$$

[12] *Isometric Array Grammars* en anglais

On peut maintenant réécrire x en y :

$$\underbrace{\begin{array}{cccccc} X & X & X & X & X & X \\ X & X & X & X & X & X \\ : & - &) & : & X & X \end{array}}_{x} \Rightarrow \underbrace{\begin{array}{cccccc} : & - & (& X & X & X \\ : & - &) & X & X & X \\ : & - & (& : & X & X \end{array}}_{y}$$

L'hypothèse d'isométrie sur les règles permet d'éviter le cas suivant qui pose un problème :

$$\underbrace{\begin{array}{ccc} X & X & X \\ : & - &) \end{array}}_{\alpha} \rightarrow \underbrace{\begin{array}{ccc} : & - & (\\ : & - &) \\ : & - & (\end{array}}_{\beta}$$

Cette règle nécessite l'ajout de trois positions et un changement de la taille du tableau. De plus, il y a un choix à faire entre les deux dérivations suivantes :

$$\underbrace{\begin{array}{cccccc} X & X & X & X & X & X \\ X & X & X & X & X & X \\ : & - &) & : & X & X \end{array}}_{x} \Rightarrow \underbrace{\begin{array}{cccccc} X & X & X & & & \\ : & - & (& X & X & X \\ : & - &) & X & X & X \\ : & - & (& : & X & X \end{array}}_{y} \textbf{ OU } \underbrace{\begin{array}{cccccc} X & X & X & X & X & X \\ : & - & (& X & X & X \\ : & - &) & : & X & X \\ : & - & (& & & \end{array}}_{y'}$$

A partir d'une grammaire et d'une configuration initiale S, Kubik définit :

- $L(SMA)$ comme l'ensemble des configurations accessibles par applications de productions contenues dans $\bigcup_i P_i$ sur S
- L_{somme} comme la *superimposition* de n ensembles L_i où chaque L_i est l'ensemble des configurations accessibles par applications de productions dans P_i sur S

Nous renvoyons le lecteur au travail original [Kub01] s'il est intéressé par la définition de l'opérateur de superimposition. Alors la propriété d'*émergence* est vraie quand :

$$\exists w \in L(SMA), w \notin L_{somme} = superimposition_i(L_i)$$

Problèmes posés Le premier problème que pose cette approche est celui de l'expression des systèmes : l'utilisation de règles de grammaires n'est pas forcément la méthode de description d'agents la plus aisée mais surtout il est difficile de savoir ce qui incarne un agent dans une configuration (un agent est un ensemble de productions).

Le second problème est celui de la décidabilité : pour dire d'une configuration qu'elle émerge, il faut pouvoir engendrer $L(SMA)$ mais aussi montrer qu'elle n'est pas dans L_{somme}. Cette affirmation repose alors sur la possibilité

de tester l'appartenance du mot à la superimposition de différents langages qui n'est pas traitée par Kubik.

Finalement, nous donnons ici une lecture critique mais qui nous semble invalider du moins partiellement les exemples de populations homogènes utilisés par Kubik. Pour une population homogène, tous les agents partagent le même ensemble de règles $\forall i P_i = P = \bigcup_i P_i$. On a alors $\forall i L_i = L(MAS)$. Par ailleurs, $superimposition(A, A) = A$ ce qui implique que $superimposition_i(L_i) = L(MAS)$. Finalement, $L(MAS) = L_{somme}$ et l'*emergence* n'a pas lieu dans le système.

Kubik fournit une tentative de formalisation de l'émergence intéressante qui repose essentiellement sur la commutation entre deux opérations : former le système avec ses composants (composition) et mettre ce système en marche (exécution).

En résumé, l'émergence de Kubik correspond au cas où des états de monde ne sont accessibles qu'à travers l'interaction de plusieurs agents. En commutant les opérations de composition et d'exécution, son travail aboutit à une forme de non linéarité qui lui permet de définir des états comme émergents.

Conclusion et Position SMA

L'ensemble des travaux qui constituent cet axe met l'accent sur la possibilité d'un gain apporté par l'interaction dans le système. Les travaux de Kubik peuvent certainement être envisagés comme une distinction entre un système *parallèle* où les différents processus ne s'influencent pas et un système *concurrent* où les interactions servent à atteindre l'objectif. L'approche d'ADELFE consiste à concentrer l'effort de conception sur les situations non coopératives qui correspondent à des interactions défaillantes ; aussi on peut voir cette proposition comme une distinction entre un système avec coopération qui pourrait être comparé avec un système sans coopération. Dans un même ordre d'idée, Searle définit différentes émergences selon le degré de collectivité entre composition et interaction.

Le problème central semble être le suivant : les approches informelles permettent de donner un principe de fonctionnement de l'émergence sans permettre directement de distinguer l'émergence et servent plus à orienter le processus de conception du système ; à l'inverse les tentatives d'approches formelles semblent très restrictives par le modèle d'agents qu'elles imposent. Dans un modèle multi-agent, cette approche correspond à l'importance du collectif, au multi de multi-agent.

2.1.8 Approche pragmatique

L'approche pragmatique a pour objectif de donner une définition de l'émergence sans avoir recours à une formulation théorique. L'idée est de donner une version de ce qu'on entend par le terme d'émergence à travers l'opinion qu'en ont les scientifiques, et de constituer un corpus d'exemples consensuels.

Nous présentons deux approches ayant choisi de définir l'émergence par l'intermédiaire d'un test ; ces deux initiatives s'inspirent de l'idée du test que Turing [Tur50] avait proposé pour définir l'intelligence. Une autre approche se fonde sur une étude d'opinion menée sur des chercheurs et étudiants lors d'une rencontre sur le thème des systèmes complexes.

Ronald Sipper et Capcarrère

Ronald Sipper et Capcarrère ont proposé un test analogue au test de Turing permettant de décider si un système produit une émergence de la même manière que le test de Turing permet de qualifier un comportement intelligent.

Test de Turing L'idée initiale de Turing était de proposer un moyen de répondre à la question "Les machines peuvent-elles penser ?"[13]. Il reformule alors le problème en un test : peut-on distinguer une machine d'un humain à travers les transcriptions textuelles de leurs réponses à une série de questions. Turing consacre une partie de son article à définir les systèmes qui forment la catégorie des machines : en particulier, il exclut les systèmes opérationnels sans que personne ne puisse décrire précisément leur mode opératoire, excluant ainsi l'homme.

Test d'émergence Le test proprement dit est formulé [RSC99] ainsi :

Conception Les entités et interactions entre entités sont décrites à l'aide d'un langage noté L1.

Observation Les observations du système global sont faites dans un langage noté L2, ce langage exclut tout terme déjà contenu dans le langage L1.

Surprise Bien que pleinement conscient des comportements élémentaires, l'observateur est surpris par ce comportement.

[13] traduit de "Can machines think ?"

L'émergence est alors le cas où l'observateur est surpris du comportement exprimé en L2 bien qu'il connaisse tout de L1.

Un exemple tiré de [RS01] concerne les techniques évolutionnistes pour la détermination de comportements de robots :

Design Le langage de conception L1 est celui des actions et perceptions basiques du robot, incluant les lectures de senseurs et l'activation de moteurs.

Observation Le langage d'observation est celui du comportement de haut niveau évolué : les robots agissent et parfois interagissent dans leur environnement (évitement d'obstacles ...). Habituellement, la fonction d'évaluation d'un algorithme génétique est alors formulée à l'aide du langage L2.

Surprise Bien que pleinement conscient des comportements élémentaires, l'observateur est surpris par la performance des robots évolués.

Il devient alors nécessaire de raffiner le terme de surprise pour raffiner celui d'émergence [RS01] :

non surprise quand il n'y a pas de surprise

surprise non surprenante la surprise est confinée à un domaine bien défini

surprise surprenante il y a réellement surprise

La difficulté de définition du test se déplace sur la notion de surprise. Afin de clarifier le sens possible de ce terme, nous en donnons un qui nous semble moins sujet à discussion tout en restant compatible avec l'idée :

non surprise un résultat déterministe obtenu grâce à un processus déterministe

surprise non surprenante un résultat déterministe obtenu grâce à un processus non déterministe

surprise surprenante un résultat non déterministe dû à un processus non déterministe

La surprise sert à la fois de notion centrale de cette proposition et d'argument principal à lui opposer : en effet, la surprise peut paraître aussi difficile à juger que l'émergence. Nous souhaitons juste finir avec un extrait de [Tur50] concernant la surprise :

> Machines take me by surprise with great frequency. This is largely because I do not do sufficient calculation to decide what to expect them to do, or rather because, although I do a calculation, I do it in a hurried, slipshod fashion, taking risks. Perhaps I

> say to myself, "I suppose the Voltage here ought to be the same as there : anyway let's assume it is." Naturally I am often wrong, and the result is a surprise for me for by the time the experiment is done these assumptions have been forgotten. ([Tur50])

> The view that machines cannot give rise to surprises is due, I believe, to a fallacy to which philosophers and mathematicians are particularly subject. This is the assumption that as soon as a fact is presented to a mind all consequences of that fact spring into the mind simultaneously with it. ([Tur50])

Test de Fabio Boschetti et Randall Gray

Boschetti et Gray [BG07] ont aussi suggéré l'utilisation d'un test d'émergence. Ce test d'émergence est destiné à détecter une des trois formes d'émergence qu'ils identifient :

Formation de motifs La formation de motifs avec l'exemple habituel du Jeu de la Vie, ces motifs n'ayant pas de pouvoirs causaux.

Emergence intrinsèque Le cas d'une propriété globale, étant influencée par des comportements individuels, qui à son tour influence les comportements individuels. L'exemple est celui d'un marché où les agents ne peuvent pas communiquer, ils prennent donc leurs décisions sur la base exclusive des propriétés globales résultantes de leurs activités locales qu'ils sont capables de percevoir. Un exemple d'émergent simple est le prix : celui-ci est construit par l'ensemble des interactions entre vendeurs et acheteurs mais c'est aussi le prix qui conditionne ces interactions.

Emergence causale D'une propriété de haut niveau sur des entités de bas niveau si cette entité exercent un contrôle sur les entités de bas niveau. La notion de causalité est alors restreinte à la notion de contrôle. L'exemple pris est celui d'un humain auquel on demande de jouer de la guitare, le contrôle est alors exercé sur une entité de haut niveau et le résultat obtenu sans interaction avec les entités de bas niveau (cellules ...) qui la composent.

Ces trois formes d'émergence correspondent alors à différents degrés de contrôle de l'entité de haut niveau sur celles de bas niveau. C'est un point de vue courant dans les travaux sur l'émergence qui consiste à lier l'intérêt d'un émergent à l'ampleur des effets dont il est la cause.

Test Le test cherche à déterminer si un processus est causal émergent. Pour cela il propose d'envisager son comportement et d'envisager si l'interaction avec le processus global (informatique) ne peut être distingué de celui d'un autre système initialement considéré comme causal émergent. Afin de distinguer ces deux systèmes, on se permet d'interagir avec eux, ce qui correspond au même cadre que celui du test de Turing.

Circularité Nous notons que leur proposition aboutit à une définition circulaire où l'émergence causale est définie comme le cas où un système émergent causal est incapable de distinguer entre un programme informatique et un autre système émergent causal. Cette définition récursive ne peut prendre de sens que si certains systèmes sont des cas de base de l'émergence causale ; les auteurs donnent un unique exemple de système émergent causal à savoir l'humain. En pratique, ce test devient alors équivalent à celui de Turing ce qui pousse à confondre les notions d'émergence et d'intelligence.

Sondage

La notion d'émergence est controversée, il ne s'y dégage pas d'opinion consensuelle ce qui a mené Christen et Franklin [CF02] à un sondage pour déterminer quelques caractéristiques de l'émergence. Au cours d'une école d'été[14], Christen et Franklin ont consulté par mail les étudiants et les chercheurs présents. Toutefois les résultats de cette enquête sont à envisager avec prudence puisque seuls 32/60 des étudiants et 6/50 des chercheurs ont répondu. Aussi l'intérêt n'est pas ici le résultat mais bien la démarche.

On note quand même que les participants ont considéré que les exemples suivants correspondent à un cas d'émergence :

- conscience
- auto-organisation
- modèle à base d'agents
- attracteurs dans les modèles non linéaires
- automates cellulaires

Ce sondage permet de situer rapidement la notion d'émergence. On note en particulier la présence des modèles à base d'agents qui suggère encore le lien entre émergence et système multi-agent. Par ailleurs, la présence de la conscience semble confirmer l'intuition que les notions de conscience, d'intelligence sont des exemples importants de l'émergence.

[14] Complex Systems Summer School, Santa Fe, 2002

La démarche correspond ici directement à un étiquetage de divers systèmes ou caractéristiques de systèmes associés à l'émergence. Ceci correspond à la finalité des deux précédentes propositions, c'est à dire constituer un corpus de systèmes émergents afin de pouvoir en dégager des caractéristiques communes.

Conclusion et Position SMA

L'objectif partagé par ces différentes propositions est de donner une définition de l'émergence par l'exemple en constituant l'ensemble des systèmes pour lesquels un test donne une réponse positive ou encore une majorité de votants le qualifie d'émergent. L'approche d'une étude d'opinion est la plus directe et consiste simplement à demander si un système est émergent. Le test fondé sur la surprise provient d'un report de la question de l'émergence à une autre notion : la surprise. L'autre test que nous avons décrit reporte l'émergence à un étalon qui sert à incarner le caractère émergent.

Dans le cadre multi-agent, la question du jugement d'exemples comme émergents n'a pas de spécificité apparente. On peut quand même remarquer que les langages du test proposé par Ronald et al. ressemblent beaucoup aux descriptions internes et externes d'un agent telles qu'on les trouve dans [DM91].

2.1.9 Bilan des grandes lignes

L'examen des multiples approches de l'émergence que l'on retrouve dans la littérature peut se couvrir à travers un ensemble d'affirmations qui n'ont pas toutes une signification claire dans le domaine multi-agent :

- L'émergence peut être la révélation de quelque chose d'existant mais aussi sa création.
- L'émergence concerne des propriétés/entités survenant sur un niveau de base tout en exerçant une causalité sur celui-ci.
- L'émergent est une propriété qui facilite la description de l'état du système ou de son futur.
- L'émergence désigne une description simple d'un comportement complexe.
- L'émergence correspond à l'interprétation d'un phénomène résultant d'une interaction entre agents et environnement.
- L'émergence correspond au cas où le système complet dépasse, pour une mesure fixée, la somme de ses composants pris séparément.

– L'émergence correspond à un avis plus ou moins consensuel, sans fondement formel regroupant tout ou partie des critères précédents.

Ce bilan pose la question du lien entre ces différentes propositions. Il est difficile de se prononcer sur la redondance ou la contradiction entre ces critères sur la base de descriptions informelles. Un travail de typologie semble intéressant pour permettre d'éclairer les similitudes et différences.

2.2 Typologies

La notion d'émergence abordée de manière directe pose beaucoup de problèmes de définitions. Une réaction possible consiste à écarter l'objectif de définition. D'un autre coté, d'autres tentent d'analyser et d'organiser les définitions disponibles pour arriver à des notions plus précises permettant un accord dans la communauté scientifique. Pour cela, la démarche générale consiste à identifier des points communs entre plusieurs définitions, à les regrouper en classes.

Les différents travaux que nous exposons ici sont des tentatives de typologies d'émergence. Il s'agit de situer les approches les unes par rapport aux autres en les décrivant dans une terminologie unique.

2.2.1 Typologie de Bedau

Mark Bedau a introduit une typologie de notions d'émergence afin d'introduire la notion d'*émergence faible*. Bedau cherche à permettre de parler d'émergence pour des systèmes où les phénomènes ont été obtenus à partir de simulations informatiques. Dans ce type de simulation, on trouve la plupart du temps des dynamiques définies à une échelle micro qui déterminent des observations et résultats au niveau macroscopique.

Nominal L'émergence nominale est le cas de propriétés d'un niveau supérieur qui ne se trouvent pas au niveau inférieur. L'exemple de Bedau est celui d'un cercle constitué de points dont aucun ne peut être qualifié de circulaire. Toutefois, il note qu'on peut encore qualifier ce cercle de résultant de points tous équidistants d'un même point (le centre). Cette définition correspond à la définition donnée par Bunge [Bun77] c'est-à-dire une propriété d'un objet complexe qui n'est pas propriété d'aucune des parties.

Forte L'émergence forte est le fait d'une propriété émergente nominale possédant des pouvoirs causaux irréductibles au sens où ils ne sont pas

l'agrégation de pouvoirs causaux du micro-niveau. Le fait de croire en cette possibilité implique alors que le niveau macro possède des pouvoirs causaux indépendants, qu'il faut alors considérer comme de nouvelles primitives dans l'étude de l'ensemble. Cet aspect de l'émergence forte est la source de la controverse autour de la notion d'émergence en philosophie et pousse Bedau à envisager une autre position qu'il considère moins fragile.

Faible L'émergence faible est alors le cas où tout le système peut s'expliquer à partir du micro-niveau mais de manière complexe. Cette manière complexe est en fait la simulation du micro niveau ce qui mène à la même définition que celle d'imprévisibilité. Bedau voit également un rapprochement avec la définition de la complexité de Kolmogorov et la notion de hasard associée : "le comportement macroscopique est aléatoire relativement à sa description microscopique". [15]

2.2.2 Typologie de l'encyclopédie de philosophie de Stanford

L'encyclopédie de Philosophie de Stanford contient un article concernant les propriétés émergentes [Sta]. On y distingue trois types d'émergence :

Émergence épistémologique L'émergence épistémologique se rapporte à ce qu'on connaît du système et non à ce qu'il est réellement mais qui ne nous est pas accessible. Il s'agit donc de décrire l'émergence comme une limitation sur la connaissance humaine concernant un système complexe. Parmi les travaux que nous avons déjà présentés, ceux de Bedau sont classés dans cette catégories ainsi que tous ceux fondés sur la notion d'imprévisibilité. En effet, la nature imprévisible d'un système n'est pas nécessaire à sa définition mais relève uniquement de l'observation de l'information disponible à celui qui cherche à le connaître et à le décrire.

Émergence ontologique L'émergence ontologique se rapporte à la constitution du système et non à ce qu'on peut en observer. La vision est alors celle d'un monde dont le niveau de base est le niveau physique et donc les niveaux supérieurs sont peuplés de composites[16]. La question centrale

[15] Nous rappelons ici qu'un nombre aléatoire au sens de la complexité algorithmique est un nombre dont la longueur est supérieure à la longueur du plus court programme permettant de l'engendrer. Ceci revient à dire que le moyen le plus concis de l'engendrer est de l'afficher, révélant ainsi une absence de structure.

[16] Système composé d'éléments physiques ou d'autres composites.

est alors celle du lien entre les niveaux et en particulier les liens de causalité éventuels entre les propriétés de ces différents niveaux.

Substance émergente Il s'agit ici d'un changement de point de vue puisqu'on qualifie d'émergent non plus des propriétés d'objets mais bien des objets. L'émergence concerne alors des objets que l'on peut qualifier de vrais composites qui possèdent des propriétés ontologiquement émergentes.

2.2.3 Typologie de Deguet Demazeau Magnin

Notre travail sur l'état de l'art de l'émergence a été d'établir quelques distinctions claires afin de distinguer les différentes approches [DDM06].

De nombreuses notions de niveaux sont mises en jeu à travers les travaux de définition de l'émergence. Toutefois, celles-ci semblent se regrouper en deux grandes catégories à savoir un distinction composants/composé et une distinction entre la description du système et son comportement. Toutefois nous avons vu que la question d'émergence change si la relation composants/composé n'est pas la relation micro macro ou encore dans le cas où le système peut percevoir son comportement (à travers l'interprétation d'une inscription). Cet état de fait oblige à poser des relations de niveaux dans un système et à expliciter les liens entre ces relations.

Conception/Observation La distinction la plus facile à identifier est celle entre les approches qui nécessitent la connaissance du système, de sa conception, de son fonctionnement interne et les approches qui ne nécessitent que l'observation du système soit l'accès à son comportement. Ce point est de fait proche de l'idée d'une description interne et externe présente dans [DM91]. En termes informatiques, la conception correspond à la description statique du système, son code alors que l'observation correspond aux traces de l'exécution du système.

Micro/Macro Il s'agit du lien entre ce que l'on considère comme l'échelle supérieure soit les niveaux globaux et locaux ou encore microscopique et macroscopique.

Lien entre ces deux relations La plupart des approches identifient les deux relations en considérant qu'on observe globalement des agents conçus localement. Pourtant l'environnement est une entité global qui est issue de la conception du système. L'existence d'une entité macroscopique qui peut

par conséquent exercer une influence se retrouve chez Muller ou encore chez Sawyer et Gilbert.

2.3 Oppositions

Forrest et Langton Dans le travail de Forrest [For90], le terme d'épiphénomène décrit un phénomène qui n'a pas d'influence sur le déroulement du système. Pour Langton [Lan90] dans le même volume, la définition est précisément axée sur l'influence mutuelle entre les deux niveaux micro et macro. Ainsi, il n'y a pas de consensus sur le besoin d'avoir une rétroaction de l'émergent sur ce qui l'a engendré.

Muller et Ronald De manière similaire Muller [Mul03] considère que l'émergence est forte dans le cas où l'observation est interne au système, c'est à dire que l'émergent est perçu par le système[17]. A l'inverse, le test d'émergence se fonde sur une surprise entre deux langages : celui de conception et celui d'observation. Si l'observation fait partie du système elle fait partie de sa conception ce qui suggère que le test sera négatif. Il semble donc qu'il y a une incompatibilité entre le test fondé sur la surprise et la notion d'émergence forte au sens de Muller.

Sawyer et O'Connor Sur le terrain de la causalité descendante, il y a également de profonds désaccords. Sawyer décrit la situation où les agents internalisent un phénomène inscrit dans une entité macroscopique. Par contre, O'Connor rappelle que la causalité descendante correspond à une influence direct du niveau supérieur et non à l'activité structurelle des entités de niveau inférieur qui constituent le niveau supérieur. Dans un cas, le niveau supérieur est une entité distincte alors qu'il est constitué du niveau inférieur dans l'autre cas.

2.4 Tables récapitulatives

Cette section a pour objectif de permettre de visualiser une partie de l'information de ce chapitre sous la forme de schémas. Il s'agit de représentations subjectives pour plusieurs raisons : tout d'abord, nous avons gommé certaines différences en regroupant les automates cellulaires et le Jeu de la

[17] On note ici que le système peut percevoir l'émergent sans rien en faire ce qui le laisserait au niveau d'épiphénomène.

Vie ; ensuite parce que de nombreux travaux décrivent plusieurs nuances d'émergence ce qui est difficile à représenter sous la forme d'une seule relation. Toutefois, ces quelques schémas nous ont aidé à distinguer les axes que nous avons choisis ce qui rend naturelle leur inclusion dans ce manuscrit.

Le premier schéma relie des auteurs ou groupes d'auteurs à des idées (cf. figure 2.1).

Le deuxième schéma correspond à un graphe où nous mettons en relation auteurs et exemples d'émergence. Ce graphe permet de réaliser que même les exemples d'émergence ne font pas l'objet d'un consensus (cf. figure 2.2). Les exemples les plus courant d'émergence sont les automates cellulaires comme le Jeu de la Vie, l'esprit ou encore le langage.

2.4.1 L'exemple du Jeu de la Vie

Un des exemples le plus régulièrement cité dans la littérature sur l'émergence est celui du Jeu de la Vie[18] qui est un automate cellulaire initialement proposé par John Conway [Gar70]. Nous étudions ici quels sont les arguments de différents auteurs concernant cet exemple.

Holland [Hol97] qualifie le Jeu de la Vie d'exemple d'émergence et s'en sert même pour tenter de définir des caractéristiques de l'émergence. En particulier, un planeur dans le Jeu de la Vie est un phénomène dont les constituants varient au cours du temps. Pour Holland, une des caractéristiques d'un phénomène émergent est la possibilité qu'il se maintienne même si ses constituants changent.

Kubik [Kub01] considère que le Jeu de la Vie n'est pas un exemple d'émergence car il ne nécessite pas d'actions conjointes de plusieurs agents. Kubik suggère un encodage du Jeu de la Vie dans le formalisme des grammaires de tableaux pour lequel son critère d'émergence n'est pas satisfait.

[18] Game Of Life ou parfois GOL.

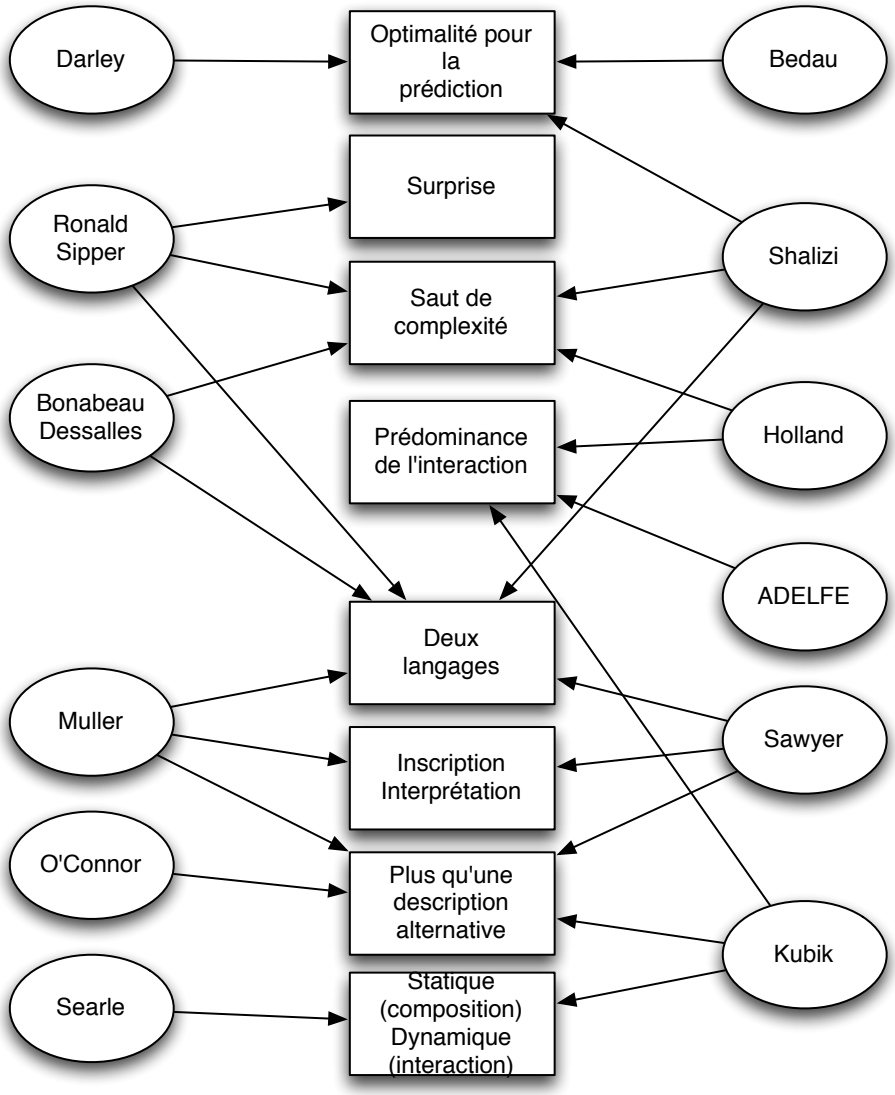

FIG. 2.1 – Liens entre les idées associées à la notion d'émergence et les différents auteurs

Bedau, Darley [Bed03] considère le Jeu de la Vie comme un exemple fondamental de système montrant de l'émergence faible. En effet, il décrit le Jeu de la Vie comme étant un exemple de système où il n'y a pas possibilité

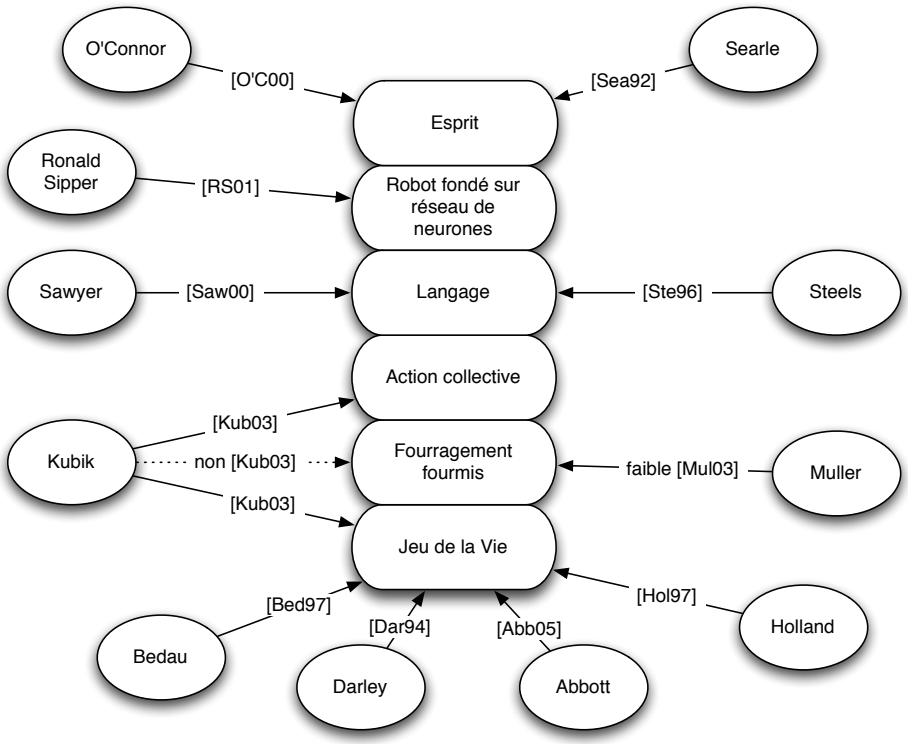

FIG. 2.2 – Liens entre auteurs et différents exemples.

de dériver l'exécution autrement que par simulation. L'exemple de base qu'il choisit est celui du R-pentamino qui est un motif dont l'évolution complexe mène à une configuration stable après 1103 pas de temps. Bedau affirme l'émergence puisqu'il ne connaît pas de moyen autre que la simulation de connaître cette configuration stable. De la même manière Darley [Dar94] considère que la dérivation de l'état stable dans le Jeu de la Vie nécessite son exécution.

Deguet Dans [Deg04], nous avons cherché à définir l'émergence comme la composante collective de la complexité d'un phénomène. Notre démarche constituait à définir la complexité d'un phénomène comme la complexité d'un détecteur de ce phénomène puis à la comparer à la complexité des entités ayant produit le phénomène. L'exemple du Jeu de la Vie et du phénomène du planeur ont servi à illustrer cette proposition à travers la

complexité du programme encodant le Jeu de la Vie comparée à celle du programme permettant d'y détecter des planeurs[19]. L'intuition sous-jacente est que l'excès éventuel de complexité entre le phénomène et le système peut être attribué aux interactions et donc au collectif. Dans le cas du Jeu de la Vie, la complexité du programme est tellement faible que la complexité de n'importe quel phénomène (détecteur) lui est supérieur.

Rabinowitz [Rab05] décrit le Jeu de la Vie comme un système émergent.

- la description de l'automate embarqué dans chaque cellule permet d'obtenir l'exécution du système
- on peut compresser une trace exécution à partir d'une boite à outils à savoir pour Life les planeurs, les blocks, l'invariance des espaces vides
- l'information mutuelle entre cette boite à outils et la description de l'automate sont hautement distinctes

Abbott [Abb06] considère le Jeu de la Vie comme un exemple d'émergence. En effet, le Jeu de la Vie est un automate cellulaire qui présente l'intérêt d'être Turing complet ce qui provoque par nécessité descendante que la non décidabilité du problème d'arrêt s'y applique. Ainsi une machine de Turing encodée dans le Jeu de la Vie peut se trouver dans des configurations où il est impossible de déterminer si l'exécution aboutira à un état stable.

2.5 Reformulation Agent

Nous essayons ici de résumer et de replacer l'ensemble des propositions en utilisant la cadre des SMA et en particulier la terminologie Voyelles. Le système est un ensemble d'agents. Un agent est une entité gérant un ensemble de valeurs variables et capable d'échanger des messages avec d'autres agents.

Tout/Parties composition Un première distinction se fait entre les propriétés des agents et les propriétés de l'ensemble des agents, l'exemple le plus simple sera pour un système constitué de deux agents la propriété "compter deux membres" est une propriété de l'ensemble sans être une propriété d'un agent. Ceci correspond à l'émergence nominale pour Bedau ou à une caractéristique système pour Searle.

[19] Il s'agit de motifs persistants périodiques.

Tout/Parties interaction Une distinction se fait entre les propriétés globales selon qu'elles peuvent être produites sans les interactions entre les différents agents. Si on a un modèle de notre système dans lequel les interactions ont été isolées, on peut alors qualifier une propriété comme émergente si on ne la retrouve que si ces interactions ont lieu. On retrouve ici les conceptions d'émergence 1 de Searle ou de Kubik qui distinguent deux cas selon la présence ou absence d'interactions. On peut alors considérer qu'une émergence nominale doit en plus être le produit de l'interaction pour être qualifiée d'émergente.

Agents et Environnement Une troisième distinction revient à savoir si le système est composé des seuls agents ou si ceux-ci cohabitent avec un environnement. Dans le cadre des systèmes multi-agent, l'existence potentielle d'un environnement permet de sortir du cadre où le système se réduit constitutivement à ses agents. Ceci rejoint l'idée de Muller[Mul03] d'un milieu d'inscription du phénomène, extérieur aux agents. Dans ce cadre, la causalité descendante est donc une propriété très courante incluant notamment tous les travaux axés autour de la notion de stigmergie où l'environnement global contraint les actions des agents qui s'y trouvent. En terme de niveaux, il s'agit de sortir de l'assimilation entre global/local et composé/composants qui sous-tend la conception du global comme relatif à l'ensemble des agents. On est alors face au cas où deux entités exercent un contrôle réciproque puisque leurs actions affectent leurs futurs respectifs ; l'environnement global et l'ensemble des agents, tous locaux.

Inscription et interprétation Dans notre modèle, dire qu'un phénomène s'inscrit dans l'environnement doit trouver une définition plus précise. Il semble que l'inscription doit être la réalisation du phénomène sous la forme d'une propriété d'un ensemble de variables. Ces variables peuvent alors être appelée les variables d'inscription du phénomène. L'inscription dans l'environnement est alors le cas où les variables d'inscription du phénomène font partie de l'environnement (ne font partie d'aucun agent). Il nous semble naturel d'exiger que la fonction d'interprétation soit une fonction partant de variables d'environnement (perçues), de variables de l'agent qui aboutissent à une modification des variables de l'agent. En cela, l'interprétation est une fonction qui provoque le changement de l'état de l'agent en fonction d'une perception mais qui est individualisée car dépendante des variables de l'agent.

2.6 Bilan sur l'émergence

Au cours de ce chapitre, nous avons envisagé l'émergence comme une propriété de haut niveau irréductible au niveau de base, comme une interaction entre deux entités vivant à des niveaux différents, comme une propriété imprévisible en toute connaissance de cause ou encore comme les possibles gains d'un collectif sur les individus. Toutefois, cette idée reste difficile à modéliser car les différentes tentatives peuvent couvrir un ensemble trop large de cas, trop étroit ou encore ne plus correspondre à l'intuition de l'émergence.

Dans la suite de ce manuscrit, nous allons nous concentrer sur une caractéristique de l'émergence soit la possibilité d'avoir un tout supérieur à la somme de ses parties.

Pistes que nous n'approfondirons pas

Nous écartons la complexité. Les mesures de complexité que nous avons évoquées n'adressent pas directement un modèle d'agents, aussi il devient nécessaire pour pouvoir les utiliser de se rapporter à leur domaine d'application qui se réduit le plus souvent à l'étude d'une séquence de symboles contenus dans un alphabet fini. Ensuite, il apparaît que la complexité associée à l'émergence est davantage celle des systèmes complexes que celle de l'informatique théorique. Dans ce cadre, les deux notions sont intimement liées comme le note Standish [Sta01]en affirmant que l'émergence est "the key ingredient that makes complex systems complex" ce qui suggère que l'émergence doit servir à la définition de la complexité et non l'inverse.

Nous n'approfondirons pas non plus la piste qui consiste à envisager l'émergence à travers l'interprétation d'un résultat de l'interaction entre agents et environnement. Nous décrirons un certain nombre des notions utiles à cette définition mais nous verrons également que la définition de ce qu'est une interprétation pose problème.

Finalement, nous n'approfondirons pas la piste suggérée pas les travaux sur la capacité à prédire le comportement futur du système car l'objectif premier de notre travail est de construire des systèmes de résolution de problèmes plus que d'étudier des systèmes existants.

Pistes que nous explorerons

La direction qui nous semble la plus intéressante est l'affirmation que le tout peut être supérieur à la somme des parties. Celle-ci est particulièrement

intéressante car cet avantage collectif est une des questions principales du domaine multi-agent. Elle est également centrée autour de la notion d'interaction qui est au coeur de l'intuition émergente.

Par ailleurs, nous explorerons ces possibilités dans le cadre d'un modèle dans lequel nous pourrons définir des vrais composites, c'est-à-dire un modèle d'agents où un système multi-agent peut être considéré comme un agent.

Ce que nous préciserons

Notre objectif est de donner un cadre dans lequel des synergies peuvent être clairement définies ainsi que l'ensemble d'expériences permettant de déterminer si ces synergies ont lieu. L'objectif est de fournir des pistes à explorer pour obtenir des gains collectifs dans un cadre de résolution de problèmes à travers une première approche fondée sur un modèle d'agents récursifs.

Pour cela, nous allons définir un modèle multi-agent ainsi qu'une instance de ce modèle : un système multi-agent de recherche à l'aide duquel nous définirons la synergie.

Chapitre 3

Modèle

Introduction

Dans le chapitre précédent, nous avons choisi de nous concentrer sur la dimension collective de l'émergence par la mise en évidence d'un tout supérieur à ses composants. Dans ce chapitre, nous exposons deux éléments principaux : un modèle d'agents hiérarchique qui nous permet de considérer agents composites et agents atomiques dans un cadre unifié et une instance de ce modèle sous la forme d'un système multi-agent pour la recherche heuristique. L'ensemble de ces deux éléments nous permet de définir de possibles synergies entre agents et d'identifier de possibles gains attribuables au collectif.

Le choix de la recherche heuristique est justifié par le fait que les heuristiques de recherche sont les premières solutions utilisées face à des problèmes inconnus et les seules présentant de bonnes performances pour de nombreux problèmes difficiles. Par ailleurs, chaque heuristique possède la capacité de trouver seule une solution ce qui rend possible l'étude séparée des composants d'un système multi-heuristique et des heuristiques qui le composent.

Nous commencerons par introduire un modèle de recherche heuristique classique dans lequel nous décrivons les notions d'espace de recherche et d'heuristique sur lesquelles se fonde notre système de recherche. Ces éléments nous servent ensuite à décrire notre système de recherche. Cette description se fera en parallèle avec celle du modèle multi-agent dont il est une instance. Nous exposerons système et modèle de l'organisation aux interactions pour ensuite décrire les agents. Finalement, nous exposerons trois synergies possibles dans notre système. Ces trois synergies correspondent aux intuitions suivantes : plusieurs heuristiques travaillant sur un même problème peuvent

améliorer la qualité d'une recherche, des recherches sur plusieurs problèmes peuvent améliorer la recherche sur le problème principal et enfin un utilisateur et un système artificiel peuvent co-construire une bonne solution.

Note Nous n'abordons pas ici la discussion habituelle consistant à opposer le théorème "no free lunch" [WM97][1] aux nombreux bons résultats observés en pratique. Nous considérons simplement que l'utilisation de la recherche heuristique est une première approche pour la plupart des problèmes pour lesquels on estime la qualité d'une solution par une fonction objectif.

3.1 Domaine d'inscription : Recherche heuristique

La recherche consiste à explorer un espace afin d'atteindre un point dont on connaît les propriétés mais pas la forme. Le caractère heuristique de la recherche correspond à l'existence d'un guide dans cette recherche, qui évite de parcourir entièrement l'espace[2]. Une heuristique de recherche consiste donc en une méthode d'exploration de l'espace qui permet de trouver une solution en changeant la forme afin d'atteindre cette propriété. Nous présentons ici la problématique générale de la recherche heuristique pour pouvoir exposer ensuite le système multi-agent correspondant. Une recherche heuristique est souvent décrite comme une succession de "generate and test" [Yao99] ; dans l'exposé de la recherche heuristique, nous suivrons cette approche.

3.1.1 Espace de recherche

Intuition Dans le cadre d'un algorithme de recherche, nous cherchons quelque chose qui satisfait un critère : un *point* . Afin de préciser le cadre de notre recherche, il faut définir quel est l'espace de ces points, c'est-à-dire l'espace dans lequel une heuristique peut se déplacer.

Le premier élément à déterminer est l'espace à explorer, l'*espace de recherche*. La notion d'espace de recherche est classique, on rencontre aussi parfois le terme d'espace d'hypothèses [CMK02].

[1] On peut traduire "no free lunch" par la "non existence d'une panacée". Ce théorème montre que toutes les méthodes de recherche heuristique se valent si on les compare sur l'ensemble des problèmes possibles.

[2] Ce type de recherche est appelé recherche en force brute ou recherche exhaustive.

3.1.2 Générateur

Intuition Un générateur produit les points voisins du point sur lequel on se trouve. Il devient naturel de qualifier de voisinage du point courant l'ensemble des points qu'il engendre par l'application du générateur. Un générateur est ce qui va nous permettre d'explorer notre espace de recherche en s'y déplaçant.

Nous définissons ici ce que nous appelons un générateur dans notre espace de recherche. A travers la littérature, on peut retrouver ce type de fonction désigné par les termes de fonction de voisinage ou encore fonction de mutation[3]. Le voisinage d'un point est l'ensemble des points produits par un générateur.

Dans le cadre d'un processus de génération et de test, les générateurs modélisent la possibilité d'engendrer de nouveaux candidats au statut de solution qui seront testés ensuite. Le type d'un générateur est $Recherche \rightarrow \mathcal{P}(Recherche)$ puisqu'il s'agit d'une fonction qui à un point associe un ensemble de points voisins.

3.1.3 Testeur

Intuition Un espace de recherche est un ensemble de points. Imaginons ces points sur une carte en deux dimensions sur laquelle nous pouvons nous déplacer de voisinage en voisinage. Un testeur est ce qui nous permet d'évaluer la qualité d'un point : son altitude. En se déplaçant dans notre carte, le testeur est un altimètre, et l'objectif de la recherche est de trouver le plus haut sommet.

Un testeur est une fonction qui à un point associe une valeur appartenant à $Valeur$. C'est cette fonction qui va nous permettre de tester l'intérêt des points produits par les générateurs.

Nous raffinons cette notion de testeur : nous considérons le cas où l'évaluation d'un point se fait par simulation. Ceci signifie qu'à un point de *Recherche* on associe un comportement contenu dans *Comportement* auquel finalement on associe une valeur dans $Valeur$. Notre testeur est donc la fonction composée d'une fonction *simulation* et d'une fonction *evaluation* .

[3]Une fonction de mutation est le terme utilisé dans les approches évolutionnistes.

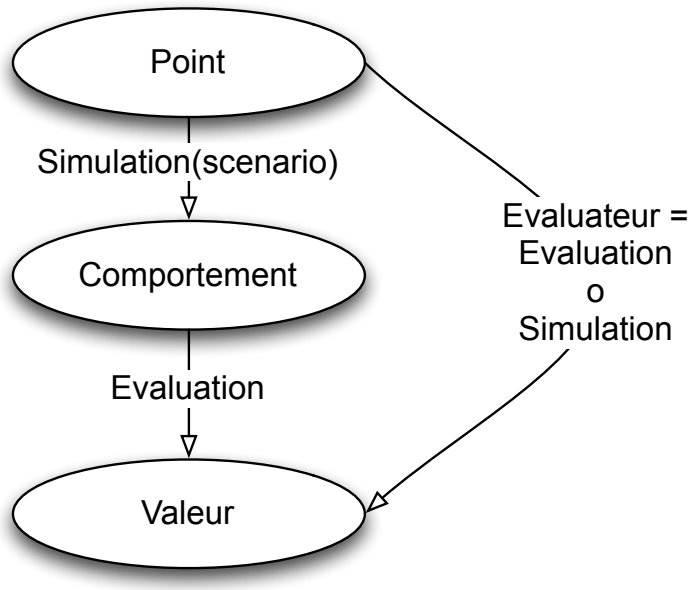

FIG. 3.1 – Un testeur comme la composée d'une fonction de simulation et d'une fonction d'évaluation

On retrouve ce type de raffinement en programmation génétique [Koz92] où contrairement à un algorithme génétique [Gol89] c'est le comportement d'un point et non un point qui est évalué.

Un testeur est donc la combinaison de deux fonctions : la première est un simulateur produisant un membre de $Comportement$ paramétré par :

- un scénario
- le point à simuler

Et d'une fonction d'évaluation associant à un comportement dans $Comportement$ une valeur dans $Valeur$.

En pratique, ceci revient à définir une fonction d'ordre supérieur à valeur dans $Valeur$ prenant finalement en paramètres :

- une fonction d'évaluation de type $Comportement \rightarrow Valeur$(ce qui rend la fonction d'ordre supérieur)
- un scénario
- un point à évaluer

Dans ce cadre, un comportement peut trouver de multiples réalisations

au niveau de l'espace de recherche[4]. Nous détaillons ce qu'est un scénario.

Scénario .

> **Intuition** L'idée de simulation est le plus souvent associée à l'évolution d'un système dans le temps. Au cours de cette évolution, certains événements peuvent être décrits plus naturellement si on les considère comme extérieurs à la fois aux agents mais également au simulateur. La séquence de ces événements constitue un scénario.

Étant donné un simulateur générique nous permettons de définir une séquence d'événements diversifiant le comportement du simulateur. Ceci permet alors à partir du simulateur générique par une application partielle d'un scénario d'obtenir une fonction de simulation pour chaque scénario.

3.1.4 Heuristique

> **Intuition** Les générateurs nous permettent de passer d'un point à un autre, le testeur nous permet de déterminer la valeur d'un point, il reste maintenant à décider comment nous allons explorer l'espace de recherche. Par exemple, l'intuition qui consiste à chercher toujours le progrès ne permet pas toujours d'atteindre l'objectif de la recherche (cf. figure 3.2). On peut donc envisager plusieurs méthodes et ce sont ces méthodes que l'on appellera heuristiques.

Étant donné un espace de recherche doté de générateurs et d'un testeur, nous pouvons maintenant définir ce qu'est une heuristique. Nous précisons la définition habituelle dans un cadre de génération et de test dépassant la définition informelle d'une méthode, inspirée d'un certain bon sens, permettant d'effectuer une recherche dans notre espace.

Dans notre modèle, nous choisissons de considérer qu'une heuristique est un processus décrivant une trajectoire dans l'espace de recherche en utilisant un ensemble de générateurs et un testeur jusqu'à s'arrêter et retourner un ensemble de points[5]. Il s'agit donc d'une fonction d'ordre supérieur prenant

[4] Il s'agit du cas où plusieurs éléments exhibent le même comportement.

[5] Le plus souvent cet ensemble est composé uniquement du meilleur point de la trajectoire.

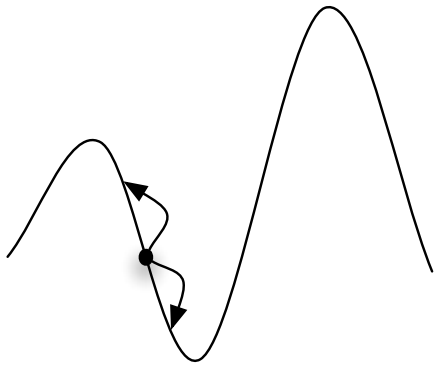

FIG. 3.2 – Une recherche cherchant à aller plus haut part vers la gauche, ce choix empêche l'heuristique de trouver le sommet : une heuristique doit parfois passer par des points de qualité moindre pour trouver un sommet.

en paramètre une fonction testeur et un ensemble de fonctions générateur ainsi qu'un point de départ. Nous considérerons des *heuristiques itératives* dont la recherche se décompose en itération que nous appelons *pas de recherche*.

3.1.5 Métaheuristiques

Intuition Il se peut qu'une recherche se décompose naturellement en plusieurs recherches : par exemple, on va pouvoir chercher une première solution de qualité moyenne puis essayer de l'améliorer ou encore chercher à imiter une solution à un autre problème pour ensuite l'adapter. Nous appellerons métaheuristique toute heuristique définie comme la combinaison d'autres heuristiques.

Une heuristique est une fonction qui partant d'un point décrit une trajectoire. L'idée d'une métaheuristique est de considérer qu'une heuristique peut en utiliser une autre voire plusieurs autres afin de construire sa trajectoire.

Dans ce contexte, une métaheuristique se définit naturellement comme une fonction d'ordre supérieur, soit une fonction prenant en paramètres d'autres fonctions heuristiques pour y faire appel durant sa recherche. On retrouve ce point de vue dans de nombreux travaux comme par exemple Osman et Laporte [OL96]"

> "A metaheuristic is formally defined as an iterative generation process which guides a subordonate heuristic"

Il est alors intéressant de noter que les heuristiques passées en paramètres peuvent utiliser un testeur et/ou des générateurs différents et donc manipuler des espaces de recherche différents.

3.1.6 Bilan

Nous avons présenté le modèle classique de la recherche heuristique fondé sur un cycle de génération et test, sur la conception d'une heuristique assimilée à une routine et des métaheuristiques comme des routines faisant appel à des sous-routines. Dans la suite, nous décrivons l'adaptation de ce cadre de recherche heuristique au domaine multi-agent avec pour objectif final la définition de synergies dans le système.

3.2 Système multi-agent de recherche heuristique

A partir de notre modèle de recherche heuristique, nous construisons maintenant notre système multi-agent de recherche. L'objectif est de permettre la définition des synergies potentielles au cours d'une recherche heuristique.

Nous décrivons notre système en partant de l'organisation des agents, cette organisation nous permet ensuite de définir les canaux possibles pour les interactions entre les différents agents du système. Une fois l'envoi de messages défini, nous pouvons décrire notre modèle d'agent en considérant un agent à travers les messages qu'il envoie et reçoit. Nous décrivons en parallèle notre modèle multi-agent et le système de recherche.

3.2.1 Organisation

Dans notre système, l'organisation est une structure qui régit le passage des messages. Notre modèle utilise une organisation imbriquée. Un agent peut en englober d'autres, que nous appellerons ses fils ; un agent est alors englobé par son père. Un seul agent n'est englobé par aucun autre, nous appelons cet agent la racine *racine* (cf. figure 3.3).

Ce type d'organisation peut aussi se représenter sous la forme d'un arbre correspondant à la relation "père de" (cf. figure 3.4). On retrouve cette idée

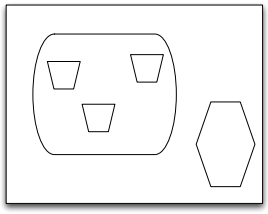

FIG. 3.3 – Plusieurs agents de différentes classes et leur organisation.

d'entités contenues dans d'autres dans le calcul des ambiants [CG98] qui propose également une organisation arborescente.

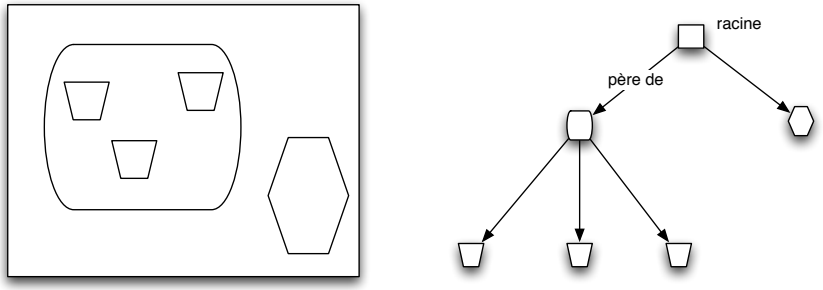

FIG. 3.4 – Représentation imbriquée et arbre correspondant.

Dans notre arbre, chaque noeud correspond à un agent : l'autre possibilité consiste à ne placer les agents que sur les feuilles, les noeuds représentant alors des groupes comme dans le modèle des SMAM [Aek99]. Notre choix permet de situer des agents à différents niveaux et d'envisager naturellement un sous-arbre comme un agent *composite* et les feuilles d'un arbre comme des agents *atomiques*.

L'organisation minimale d'un système de recherche heuristique consiste en un agent effectuant la recherche heuristique appelé agent heuristique et noté **Heuristique** qui est alors la racine et le seul agent de notre organisation (cf. figure 3.5).

FIG. 3.5 – Un seul agent heuristique

Afin de distinguer l'agent de recherche heuristique du système complet, nous introduisons un agent dit *agent problème* **Problème** . Cet agent a pour rôle de définir le problème à explorer et d'initier une recherche concernant ce problème chez l'agent heuristique. La nouvelle racine est donc cet agent problème et possède un unique fils : l'agent heuristique (cf. figure 3.6).

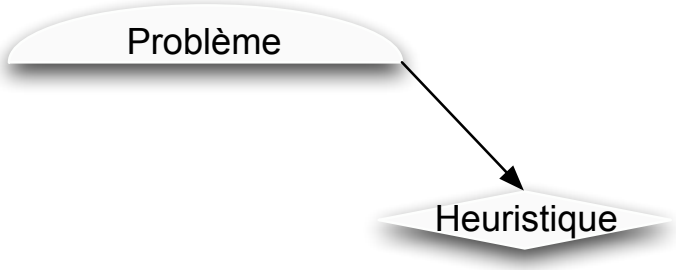

FIG. 3.6 – Un agent problème et un agent heuristique fils

3.2.2 Interactions

Les interactions de notre système sont des envois de messages asynchrones entre agents. Cela signifie que l'envoi de messages ne bloque pas l'agent qui continue ensuite son exécution et que la réception de messages se fait à travers une file de messages : l'ordre de traitement suit l'ordre de réception.

Un message est donc un contenu associé envoyé d'un expéditeur à un destinataire, c'est-à-dire un triplet $\langle$ expediteur,contenu,destinataire $\rangle$. Dans le domaine multi-agent, notre modèle correspond à ce que Ferber [Fer95] appelle des agents *purement communiquants*.

Type de message Nous utilisons un système de type simple pour les contenus de messages. Il s'agit de tuples constitués de valeurs avec une possibilité de nommage. Ainsi on peut définir le type *Rapporter* dont le contenu est un tuple composé d'une séquence de couples ⟨ hypothese,valeur ⟩ comme ⟨ [⟨ hypothese,valeur ⟩] ⟩ . En utilisant le nom *trajectoire*, on obtient ⟨ trajectoire :[⟨ hypothese,valeur ⟩] ⟩ .

L'appartenance d'un message à un type se décide par le nom de ce type est non par sa structure. Ainsi on peut distinguer deux messages de structures ⟨ ⟩ par le nom de type utilisé bien qu'ils partagent la même structure.

Nous utiliserons la notation suivante pour un type de message de nom *Initier* enregistrement d'un champ testeur contenant une fonction de *Recherche* vers $Valeur$, d'un champ générateurs contenant une liste de générateurs et d'un champ départ contenant un point.

Initier : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩

Destinataires Dans le cadre de l'organisation que nous avons définie, nous définissons trois possibles destinataires pour un message :

boucle Le message est envoyé à l'agent expéditeur.

père Le message est alors envoyé au seul père de l'agent expéditeur, il est perdu si l'expéditeur est la racine de l'organisation.

fils Le message est envoyé aux fils de l'agent expéditeur, il est perdu si l'expéditeur n'a pas de fils.

Interface Une interface agent correspond à l'ensemble des types de messages que l'agent qui l'implémentera est susceptible de recevoir soit son *interface d'entrée* et à l'ensemble des types de messages que cet agent peut envoyer soit son *interface de sortie* . Nous présentons les interfaces des deux classes d'agents mentionnés auparavant à savoir un agent heuristique et un agent problème.

Interface d'agent heuristique

Interface Heuristique

Entrées

Initier : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩
Stopper : ⟨ ⟩
Boucler : ⟨ ⟩

Sorties

Rapporter : ⟨ [⟨ point :Point,valeur :Valeur ⟩] ⟩
Boucler : ⟨ ⟩

Nous donnons ici l'usage de ces types de messages. Initier la recherche consiste à commencer à naviguer dans l'espace de recherche à la recherche d'une bonne solution, stopper signifie arrêter cette recherche. Si l'agent effectue sa recherche il ne récupère plus de messages ; c'est la justification du type de message Boucler : l'agent décompose sa recherche en *pas* de recherche afin de pouvoir traiter des messages entre ces pas. Ceci permet de maintenir l'agent à l'écoute pendant sa recherche.

Interface d'agent problème

Interface Problème

Entrées

Rapporter : ⟨ [⟨ point :Point,valeur :Valeur ⟩] ⟩

Sorties

Initier : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩
Stopper : ⟨ ⟩

On voit ici que l'interface de l'agent problème est principalement le miroir de l'interface de l'agent heuristique. Comme nous n'avons encore défini que deux types d'agents, il est naturel que le premier puisse recevoir les types de messages que le second peut envoyer et inversement.

3.2.3 Agents

La description des précédentes facettes d'un système multi-agent nous permet de faire le point ici sur ce qu'est un de nos agents. Il s'agit d'un noeud dans un arbre ayant ainsi un père et des fils. L'agent est doté d'une file de réception de messages et il peut en envoyer.

Dans la littérature, la notion d'agent a de multiples définitions dont on peut extraire un squelette quasiment consensuel : un agent est une entité dotée de capacités de :

- Perception
- Décision
- Action

Dans notre modèle, l'enchaînement de ses trois étapes correspond à la boucle principale commune à tous les agents (cf. algorithme 1)[6].

Algorithme 1 : Boucle principale d'agent

```
   Data : mémoire
1  begin
2      ⟨ memoire,messages ⟩ = ALaNaissance(memoire);
3      EnvoyerMessages(messages);
4      while Vivant do
5          reçu = ReceiveMessage();
6          if type(reçu) == Meurt then
7              Vivant = faux;
8          end
9          else
10             ⟨ memoire,AEnvoyer ⟩ = TraiterMessage( ⟨
               memoire,recu ⟩ );
11             EnvoyerMessages(AEnvoyer);
12         end
13     end
14     ⟨ memoire,messages ⟩ = ALaMort(memoire);
15     EnvoyerMessages(messages);
16 end
```

Nous précisons la sémantique sous-jacente. La fonction appelée à la naissance de l'agent permet un traitement initial de la mémoire ou encore d'envoyer des messages. La boucle principale de l'agent est conditionnée par la

[6] Un résumé des notations algorithmiques est fourni en annexe.

variable *vivant* . Cette variable est initialement *vrai* et n'est modifiée que par la réception d'un message de type **Meurt** . Le régime habituel d'un agent consiste à traiter le message reçu par l'application d'une fonction prenant en entrée le message reçu et l'état courant de la mémoire à la réception et renvoyant les messages à envoyer ainsi que le nouvel état de la mémoire de l'agent.

Nous faisons les remarques suivantes :

- La boucle principale traite de messages et non de perceptions, ceci signifie que toute action ou perception est modélisée par un message.
- Il y a séparation entre la partie fonctionnelle de l'agent et sa partie mémorielle. L'appel d'une fonction suggère que le même message entraînera le même comportement de la part de l'agent. Toutefois, le fait que l'agent ait une mémoire à disposition lui permet de distinguer des situations et donc d'adapter son comportement. L'agent procède donc par applications successives de fonctions de sa mémoire courante et du message perçu.
- Les messages sont traités de manière séquentielle ce qui peut paraître une limitation. Toutefois, cela permet de définir de manière claire ce qu'est une trace d'agent à savoir une séquence d'états mémoriels dont les transitions sont étiquetées par les messages reçus. Les traitements simultanés doivent mettre en jeu plusieurs agents.
- L'exécution d'un agent se fait en réponse à la réception de messages. Il doit donc y avoir un message initial qui correspond au point d'entrée d'un programme habituel.

Le définition d'un type d'agent se réduit à la manière dont il traite les messages qu'il reçoit et en particulier aux messages qu'il envoie en réponse. Le détail de cette description passe par la définition du type de la mémoire et la définition d'une fonction de traitement par type de messages de l'interface d'entrée.

Mémoire La définition du type de mémoire d'un agent se fait selon la même notation que celle utilisée pour la définition du type d'un contenu de message. Il s'agit donc de définir un tuple avec une possibilité de nommage. Ainsi on pourra noter le type de la mémoire d'un agent capable de mémoriser les coordonnées cartésiennes d'un point par le type $\langle$ x :Flottant,y :Flottant,z :Flottant $\rangle$ où Flottant est un type existant.

Comparaison avec le modèle d'acteurs et le langage Erlang Le modèle d'acteurs [HBS73] correspond à une des premières approches d'en-

tités communiquant par envoi de messages, le langage Erlang [Erl, Arm03] est fortement inspiré par ce modèle. L'entité élémentaire du système est soit un acteur (modèle d'acteurs) soit un processus (Erlang) soit pour nous un agent. Nous partageons plusieurs caractéristiques avec ces modèles : les entités n'interagissent que par envoi de messages, elles ne partagent pas de structure de données et adoptent une vision fonctionnelle du traitement effectué par un agent. Des différences cosmétiques existent concernant les ordres de réception de messages ou encore la garantie de leur livraison. Les différences importantes de notre modèle permettent de pointer des notions utiles dans la discussion sur l'émergence :

- Erlang permet de modifier la fonction de traitement d'un agent en cours d'exécution ce qui oblige à suivre la fonction de traitement ainsi que les données d'un agent pour obtenir la trace d'un agent. Dans notre modèle, la trace d'un agent ne nécessite que les changements de mémoire.
- Les modèles d'acteurs et d'Erlang présente une structure à plat, la structure hiérarchique de notre modèle permet un adressage relatif, la définition naturelle de sous-systèmes ainsi qu'une notion de niveaux.

En résumé, nous conservons l'idée d'un traitement fonctionnel du modèle d'acteurs tout en permettant une mémoire plus élaborée que le simple ensemble d'accointances permis par le modèle d'acteurs, point que nous partageons avec l'approche d'Erlang.

Comparaison avec le modèle MAGIQUE Le modèle MAGIQUE [Mag, BM97] correspond à la description de systèmes Multi-AGent hiérarchIQUEs. Il est fondée sur une organisation hiérarchique dans laquelle les feuilles que nous avons appelées agents atomiques sont des *spécialistes* et les noeuds de l'arbre sont des *superviseurs*. Les traitements de messages sont effectués par des *capacités* dont l'agent est équipé, lesquelles sont similaires à nos fonctions de traitement. Toutefois, un agent MAGIQUE n'a pas de mémoire, les données d'un agent correspondent dès lors à la réunion des données de ses capacités. Ces capacités ne sont donc pas des fonctions mais plutôt des données associées à des fonctions. Ceci a un impact sur le mode d'exécution d'un agent : les traitements de différents messages sont parallèles dans un agent MAGIQUE alors qu'un de nos agents doit attendre une éventuelle modification de sa mémoire pour en tenir compte dans les traitements suivants ; ce qui définit un mode séquentiel. D'un point de vue organisationnel, en dehors des liens hiérarchiques, MAGIQUE permet de maintenir des liens d'accointances qui sont une possibilité d'interaction dont

n'est pas doté notre modèle. Finalement, si certaines caractéristiques importantes diffèrent, le modèle MAGIQUE est sans doute le modèle explicitement multi-agent le plus proche de notre proposition.

Voyelles L'approche VOYELLES [Dem01] consiste à envisager un système multi-agent comme une population organisée d'agents en interaction, situés dans un environnement partagé. Le principe de récursivité consiste à considérer qu'un système multi-agent est lui-même un agent permettant ainsi de les envisager de la même manière. Notre modèle est une réalisation de ces principes où nous avons permis d'englober des agents à l'intérieur d'un autre. Dans notre arbre, chaque noeud est un agent, et chaque sous arbre est un système multi-agent avec lequel on interagit par sa racine : pour le reste du système, il n'y a donc pas de différence entre un agent atomique et un agent composite. Nous n'avons pas donné de définition explicite de l'environnement, toutefois si on considère l'environnement comme l'endroit où un agent est plongé, le père d'un agent peut être vu comme son environnement avec lequel il interagit par les messages qu'ils échangent.

Agent Problème

Un agent problème correspond à la gestion de la recherche heuristique pour un problème donné. Ce type d'agent doit mémoriser un problème défini par un espace de recherche, des générateurs sur cet espace et un testeur. Ce type d'agent doit également maintenir l'information utile sur la recherche déjà effectuée en gardant les bonnes solutions trouvées auparavant.

Mémoire Le type de la mémoire d'un agent problème correspond à son rôle dans l'exécution du système. L'agent problème maintient l'information de définition du problème qu'il représente à savoir un testeur, un ensemble de générateurs et un descripteur de l'espace de recherche. Il garde également les bonnes solutions trouvées jusque là sous la forme d'une séquence de couples $\langle$ Point,Valeur $\rangle$: sa *trajectoire* . Ceci donne le type de la mémoire soit $\langle$ probleme : $\langle$ testeur :Testeur,generateurs :[Generateur],point :Point $\rangle$,trajectoire :[$\langle$ Point,Valeur $\rangle$] $\rangle$.

Fonctions de traitement Le comportement de notre type d'agent se définit à travers les fonctions de traitement qu'il appliquera à réception des messages. Nous n'avons défini qu'un seul type de messages dans l'interface d'entrée. Le traitement effectué consiste à conserver le point du rapport.

Algorithme 2 : Traitement par un agent Probleme d'un message Rapporter

```
  Data : message, un message de type Rapporter
  Data : memoire, la memoire courante
1 begin
    /* Calcul de la nouvelle structure stockant le meilleur
       point                                                 */
2   extension = ajoutdroite(memoire.solutions, message.point);
    /* Mise à jour de la mémoire sans envoi de message       */
3   return ( ⟨ ⟨ memoire.probleme,extension ⟩ ,[] ⟩ );
4 end
```

Notre type d'agent problème stocke le meilleur point issu de chaque rapport de recherche qu'il reçoit sans envoyer de messages. Si le point est meilleur que ce qu'on trouve dans la trajectoire, le point et sa valeur sont concaténés à la trajectoire.

La liste de messages vide [] indique que l'agent n'envoie pas de messages.

Agent Heuristique

Un agent heuristique est un agent qui cherche une solution au problème de agent problème. La manière dont il effectue cette recherche dépend de l'heuristique choisie ; nous décrirons plus tard plusieurs heuristiques agentifiées. Un agent heuristique va effectuer un pas de recherche pour chaque message Boucler. L'intuition de l'agentification d'une heuristique est alors de permettre l'écoute de messages provenant de l'extérieur entre les pas recherche successifs.

A titre de premier exemple, nous décrivons un type d'agent simple que nous appellerons *Descente* consistant à choisir un générateur parmi l'ensemble de générateurs puis à partir d'un point de départ, à se déplacer vers un meilleur voisin tant qu'il y en a un.

Mémoire La mémoire d'un agent de descente contient un testeur, un générateur ainsi qu'une séquence de points. On la définit donc par ⟨ testeur :Testeur,generateur :Generateur,trajectoire :[⟨ Point,Valeur ⟩] ⟩ .

Fonctions de traitement La première fonction de traitement (cf. algorithme 3) correspond à l'initialisation de la mémoire avec le tuple constitué

du testeur, du premier générateur de la séquence envoyée et de la séquence de points ne contenant que celui de départ. Un message Boucler est également envoyé par l'agent à lui-même.

Algorithme 3 : Traitement par un agent Descente d'un message Initier

Data : message, un message de type Initier
Data : memoire, la memoire courante

Data : message, un message de type Initier
Data : memoire, la memoire courante
1 **begin**
2 memoire = ⟨ message.testeur,premier(message.generateurs),[message.depart] ⟩ ;
3 renvoyer (⟨ nouvelle,[Boucler()] ⟩);
4 **end**

Le traitement d'un message Boucler (cf. algorithme 4) permet d'effectuer un pas de recherche. Si l'agent a progressé, la recherche se poursuit, sinon l'agent envoie un rapport de recherche.

Le message Stopper (cf. algorithme 5) correspond à l'interruption prématurée de la recherche. En conséquence, l'agent de descente envoie un rapport de recherche.

Si l'agent a dans sa file un message de type Boucler, le message Stopper provoque l'émission d'un rapport mais n'empêche pas la poursuite de la recherche.

3.3 Synergies

Nous avons maintenant fini de décrire notre modèle multi-agent ainsi que deux classes agents. Nous présentons trois synergies afin d'explorer une possible supériorité collective. Pour chacune de ces synergies, nous précisons sa signification, l'extension nécessaire de notre système de recherche et les moyens d'en évaluer l'impact.

3.3.1 Synergie d'heuristiques

Pour un problème donné, modélisé par un testeur et un espace de recherche dotés de générateurs, nous cherchons à modéliser la notion de sy-

Algorithme 4 : Traitement par un agent Descente d'un message Boucler

```
   Data : message, un message de type Boucler
   Data : memoire, la memoire courante
 1 begin
      /* calcul du voisinage à partir du générateur contenu
         dans la mémoire                                    */
 2    voisins = memoire.generateur(courant);
      /* calcul des valeurs des voisins                     */
 3    evalues = {};
 4    forall voisin ∈ voisins do
 5       evalues = union(evalues, ⟨ voisin,memoire.testeur(voisin) ⟩ );
 6    end
      /* récupération du meilleur                           */
 7    meilleur = meilleur(evalues);
 8    if meilleur.valeur ≥ courant.valeur then
         /* la nouvelle mémoire a un nouvel élément en fin de
            trajectoire, on envoie un message de bouclage   */
 9       memoire = memoire   avec   trajectoire =
         ajoudroite(trajectoire,meilleur);
10       return ⟨ memoire,[Boucler()] ⟩
11    end
12    else
         /* on envoie un rapport contenant la trajectoire de
            recherche                                       */
13       rapport = Rapporter(pere,meilleur(memoire.trajectoire));
14       return ⟨ memoire,[rapport] ⟩
15    end
16 end
```

Algorithme 5 : Traitement par un agent Descente d'un message Stopper

Data : message, un message de type Stopper
Data : memoire, la memoire courante
1 **begin**
2 rapport = Rapporter(pere,meilleur(memoire.trajectoire));
3 **return** ⟨ *memoire,[rapport]* ⟩;
4 **end**

nergie entre agents heuristiques. Il s'agit alors de vérifier l'apport de l'interaction dans le processus de recherche.

Dans le cadre du système de recherche, il s'agit de travailler sur les interactions entre plusieurs agents heuristiques effectuant une recherche sur le même espace de recherche doté du même testeur. Il s'agit donc d'interactions au niveau de l'arbre constitué d'un agent problème et de ses plusieurs agents heuristiques.

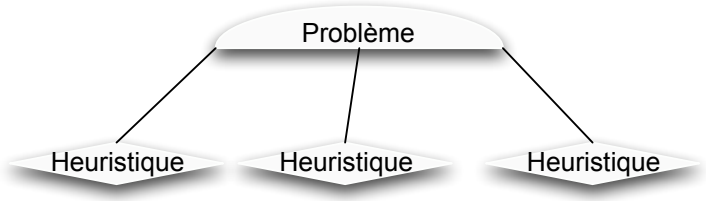

Fig. 3.7 – Un agent problème et plusieurs agents heuristiques

L'interaction que nous utilisons est le partage entre agents heuristiques d'un point de l'espace de recherche. Pour cela, on enrichit l'interface d'un agent heuristique en lui permettant d'envoyer et de recevoir un point **Partager** : ⟨ point :Point,valeur :Valeur ⟩ . L'envoi de ce point se fait vers l'agent problème qui le relaie aux agents heuristiques si le point constitue un progrès dans la recherche. L'interface de l'agent problème est donc également modifiée.

Interface Heuristique

Entrées

Initier : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩
Stopper : ⟨ ⟩
Boucler : ⟨ ⟩
Partager : ⟨ point :Point,valeur :Valeur ⟩

Sorties

Rapporter : ⟨ [⟨ point :Point,valeur :Valeur ⟩] ⟩
Boucler : ⟨ ⟩
Partager : ⟨ point :Point,valeur :Valeur ⟩

Interface Probleme

Entrées

Rapporter : ⟨ [⟨ point :Point,valeur :Valeur ⟩] ⟩
Partager : ⟨ point :Point,valeur :Valeur ⟩

Sorties

Initier : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩
Stopper : ⟨ ⟩
Partager : ⟨ point :Point,valeur :Valeur ⟩

Afin de permettre deux modes d'exécution, on ajoute un nouvel élément à la mémoire de l'agent problème selon que l'agent relaie ou pas les progrès aux heuristiques : la valeur booléenne *partage*. Ainsi le type de la mémoire d'un agent problème devient ⟨ probleme : ⟨ testeur :Testeur,generateurs :[Generateur],point :Point ⟩ ,trajectoire :[⟨ valeur,[Point] ⟩],partage :Booleen ⟩

La fonction de traitement d'un message Partager consiste donc à retenir le point si celui-ci est un progrès et à le partager si *partage* est vrai (cf. algorithme 6).

Pour un agent heuristique, il reste à définir quand un point est partagé et la manière dont est reçu un message **Partager** : ⟨ point :Point,valeur :Valeur ⟩ . Ces deux points seront précisés dans la description des heuristiques

Algorithme 6 : Traitement par un agent Probleme d'un message Partager

```
Data : message, un message de type Partager
Data : memoire, la memoire courante
begin
    meilleur = meilleur(memoire.trajectoire);
    /* Si le meilleur est au moins aussi bon, et que le
       partage est activé, transmettre le message aux fils
       */
    if (meilleur.valeur ≤ message.valeur) et (memoire.partage) then
        extension = ajoutdroite(memoire.solutions,message.point);
        memoire = memoire    avec    trajectoire = extension;
        return ⟨ memoire,[Partager(fils,message)] ⟩
    end
    /* Dans le cas contraire ne rien faire                    */
    else
        return ⟨ memoire,[] ⟩
    end
end
```

dans le chapitre suivant.

Évaluation Afin d'évaluer l'intérêt de cette possibilité, nous étudions la performance du système de recherche heuristique selon le nombre et la classe des heuristiques et selon la mise en interaction de ces heuristiques par la valeur de *partage*. Le critère d'évaluation est la qualité de la meilleure solution en fin de recherche.

Chaque agent heuristique stoppe sa recherche s'il a atteint un quota d'appels au testeur ou s'il a effectué un nombre fixé d'appels au testeur sans progresser.

3.3.2 Synergie de problèmes

Nous avons modélisé un problème par un agent responsable de la progression de la recherche vers de bonnes solutions, le degré de qualité d'une solution étant obtenu par un testeur. La possibilité d'une synergie de problème correspond au gain éventuel d'une interaction entre agents responsables de différents problèmes.

La levée de la restriction à un seul problème mène à certaines précisions : tout d'abord il faut définir le problème principal dans l'ensemble de problèmes explorés. Dans notre approche, les problèmes voisins sont des problèmes où :

- la fonction d'évaluation est changée
- le scénario est changé

Nous réduisons donc la différence de problèmes à une différence de testeurs, l'espace de recherche et les générateurs étant conservés.

D'un point de vue organisationnel, la synergie de problèmes consiste donc à envisager le système avec plusieurs agents problème et non plus un seul comme vu précédemment. Afin de conserver une structure d'arbre, cela nous amène à définir un agent qui soit le père de tous les agents problèmes. Cet agent sera appelé **Automates** . L'agent **Automates** a une mémoire de type $\langle$ principal :Testeur,couples :[$\langle$ testeur :Testeur,meilleur : $\langle$ Point,Valeur $\rangle$ $\rangle$],voisin :($Testeur \rightarrow Testeur$) $\rangle$ qui lui permet de conserver l'information du testeur principal ainsi que le meilleur point pour chaque testeur.

Au niveau interaction, il convient également d'enrichir l'interface d'un agent problème pour que celui-ci puisse interagir avec **Automates** .

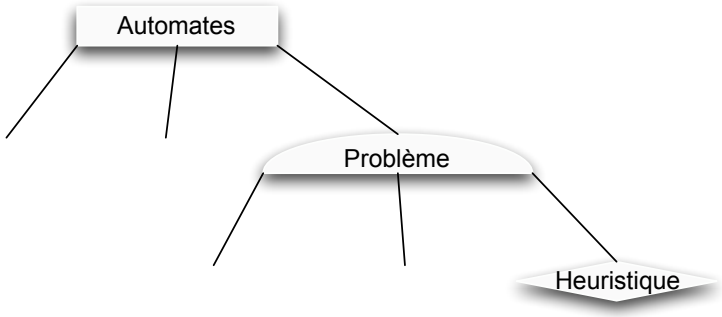

FIG. 3.8 – Nouvelle organisation avec l'introduction d'un agent supervisant la recherche automatique.

Nous introduisons à cet effet le type de message **Mandater** : ⟨ actif :Testeur,depart :Point ⟩ envoyé par l'agent Automates vers ses fils. A sa réception, l'agent responsable du problème se charge de la recherche. En réponse, l'agent problème enverra un message **RendreCompte** : ⟨ meilleur : ⟨ point :Point,valeur :Valeur ⟩ ,testeur :Testeur ⟩ contenant le résultat de la recherche. Cela correspond donc à un enrichissement de l'interface d'un agent problème (cf. algorithme 7) et à la création d'une interface de l'agent Automates.

Interface Probleme

Entrées

Rapporter : ⟨ [⟨ point :Point,valeur :Valeur ⟩] ⟩

Partager : ⟨ point :Point,valeur :Valeur ⟩

Mandater : ⟨ actif :Testeur,depart :Point ⟩

Sorties

Initier : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩

Stopper : ⟨ ⟩

Partager : ⟨ point :Point,valeur :Valeur ⟩

RendreCompte : ⟨ meilleur : ⟨ point :Point,valeur :Valeur ⟩ ,testeur :Testeur ⟩

Interface Automates

Entrées

RendreCompte : ⟨ meilleur : ⟨ point :Point,valeur :Valeur ⟩ ,testeur :Testeur ⟩

Sorties

Mandater : ⟨ actif :Testeur,depart :Point ⟩

Au niveau de la définition d'agent, il convient de spécifier comment un agent problème réagit aux nouveaux messages par la définition d'une nouvelle fonction de traitement relayant aux fils le nouveau point de départ (cf. algorithme 7).

Algorithme 7 : Traitement par un agent Probleme d'un message Mandater

Data : message, un message de type Mandater
Data : memoire, la memoire courante

1 **begin**
  `/* Si je suis le responsable de la recherche sur ce problème je relaie ce point de départ */`
2 **if** *message.actif == memoire.probleme.testeur)* **then**
3 **return** ⟨ *memoire,[Initier(fils,* ⟨ *memoire.probleme.testeur,memoire.probleme.generateurs,message.depart* ⟩ *)]* ⟩
4 **end**
  `/* Si ce n'est pas mon problème, on stoppe. */`
5 **else**
6 **return** ⟨ *memoire,[Stopper(fils)]* ⟩
7 **end**
8 **end**

Évaluation L'évaluation de la possibilité d'une synergie entre problèmes se fait à travers l'étude de l'impact d'une recherche sur un problème voisin avant d'effectuer la recherche principale. Le résultat de cette première recherche, c'est à dire la solution est alors fournie comme point de départ à la recherche principale. Nous évaluons donc à quel point cet autre problème favorise la découverte d'une bonne solution à notre problème principal.

Pour chaque problème voisin, nous étudierons le gain de performance sur le problème principal.

3.3.3 Synergie agents-utilisateur

Nous avons considéré un système de recherche où nous avons permis à plusieurs agents d'échanger des solutions au même problème ou encore de fournir la meilleure solution d'un problème à un autre. La possibilité d'une synergie agents-utilisateur correspond au cas où une solution est trouvée grâce à un apport de l'homme conjoint à un apport des agents.

A un niveau organisationnel, cette possibilité nécessite l'introduction de l'agent **Système** créant l'articulation entre l'ensemble des agents artificiels et l'humain à travers un agent **Utilisateur** (cf. figure 3.9).

Interface Utilisateur

Entrées

PrincipalMeilleur : ⟨ meilleur :Point ⟩

Sorties

Comportement : ⟨ comportement :Comportement ⟩

Interface Systeme

Entrées

Comportement : ⟨ comportement :Comportement ⟩

PrincipalMeilleur : ⟨ meilleur :Point ⟩

Sorties

Comportement : ⟨ comportement :Comportement ⟩

PrincipalMeilleur : ⟨ meilleur :Point ⟩

Cet agent effectue le relais de messages entre **Utilisateur** et **Automates** . **Utilisateur** consiste en une interface qui permet à l'utilisateur d'envoyer des comportements à **Système** .

L'envoi de comportement provoque la création d'un agent problème par **Automates** dont l'objectif de recherche est d'imiter le comportement

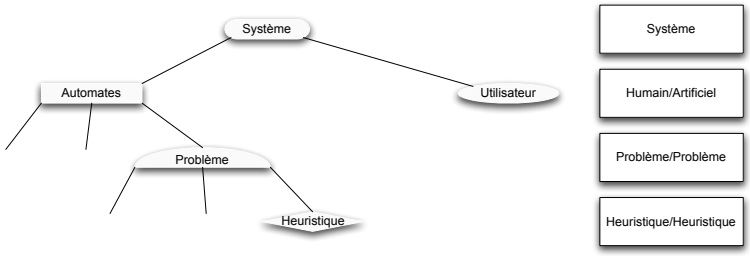

FIG. 3.9 – Version finale de l'organisation en arbre

fourni. Ceci donne lieu à la définition d'un agent de problème particulier : un agent d'imitation. Le but d'un agent d'imitation est d'imiter le comportement fourni afin de pouvoir ensuite explorer à partir des points imitant le comportement fourni. (cf. figure 3.10).

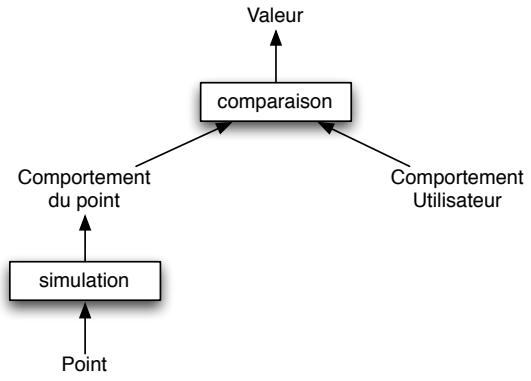

FIG. 3.10 – Un point est simulé puis le comportement produit comparé à celui fourni en référence.

Nous précisons la forme d'un comportement. Il s'agit d'une séquence de couples entrée/sortie où entrée et sortie sont deux tuples de valeurs nommées. On peut par exemple considérer [⟨ [a=1,b=0],[c=1] ⟩ , ⟨ [a=1,b=1],[c=0] ⟩ ,tuple[a=0,b=0],[c=0]] (cf. figure 3.11).

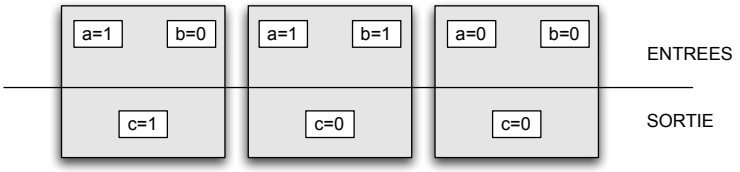

FIG. 3.11 – Un exemple de comportement.

Évaluation Le mode d'évaluation que nous utilisons pour évaluer l'apport du comportement fourni est de tenter de l'imiter en premier lieu puis de fournir le meilleur imitant comme point de départ à la recherche principale.

La synergie est alors évaluée par la comparaison entre :
- La qualité du comportement fourni
- La qualité de la solution produite par le système artificiel
- La qualité de la solution obtenue avec imitation préalable

3.4 Situation

3.4.1 Recherche heuristique multi-agent

L'idée d'un système de recherche heuristique multi-agent se retrouve dans d'autres travaux. Milano et Roli [MR04] ont proposé un cadre dans lequel ils formulent un grand nombre de métaheuristiques à l'aide d'agents utilisant d'autres agents. Il existe deux différences principales avec nos travaux :
- Pour Milano et Roli, chaque recherche aboutit sans être interruptible, aussi la décomposition est équivalente à une décomposition fonctionnelle. Un agent ayant commencé une recherche ne s'arrête pas avant la fin de cette recherche.
- Aucun moyen systématique d'évaluation de l'apport d'une interaction ou d'un agent n'est fourni.

Ces travaux et les nôtres suggèrent que l'expression d'heuristiques comme des agents peut servir à une étude "sociale" de ces différentes heuristiques mais aussi à la suggestion de nouvelles combinaisons plus souples que la subordination d'une heuristique à une autre.

Notre système s'intègre dans le domaine plus large de l'étude de la recherche coopérative où plusieurs heuristiques coopèrent afin de trouver de bons résultats. Un exemple de ce type de travaux [BCK05] combine un algorithme génétique et une recherche tabou travaillant sur le même ensemble

de solutions afin de résoudre un dérivé du problème du voyageur de commerce. Certaines caractéristiques de notre système nous distinguent de ces travaux :

problèmes voisins Nous avons modélisé l'idée de problèmes voisins qui correspondent à des versions simplifiées ou modifiées du vrai problème adressé par la recherche ce qui nous a mené à envisager une synergie de problèmes. Cette idée se retrouve dans de nombreux travaux d'apprentissage incrémentiel comme [GM97] mais pas à notre connaissance dans le domaine des heuristiques coopératives.

recherche de fonction Nous avons décrit un cadre de recherche de fonctions, qui permet de découpler le comportement d'une solution de sa valeur.

3.5 Bilan

Dans ce chapitre, nous avons introduit notre modèle de systèmes multiagent. Nous avons construit ce modèle autour de deux principes : un agent fonctionnel dont la vie est une succession d'applications de plusieurs fonctions de traitement et une organisation hiérarchique permettant d'envisager systèmes composites et atomiques de manière uniforme. Cette vision hiérarchique introduit également une notion de niveau correspondant au degré de composition du système.

Nous avons également décrit un système de recherche heuristique multiagent en modélisant une heuristique par un agent, ce qui permet naturellement d'envisager une métaheuristique comme un SMA. Cette modélisation permet pour des heuristiques itératives d'envisager une communication en cours de recherche, ce qui nous permet d'envisager de distinguer deux cas pour une population d'agents selon qu'ils échangent ou non de l'information pendant leurs recherches individuelles.

L'ensemble nous a permis de définir des synergies possibles entre heuristiques, problèmes ou encore entre l'utilisateur et le système. Nous allons maintenant chercher à savoir si ces synergies ont bien lieu.

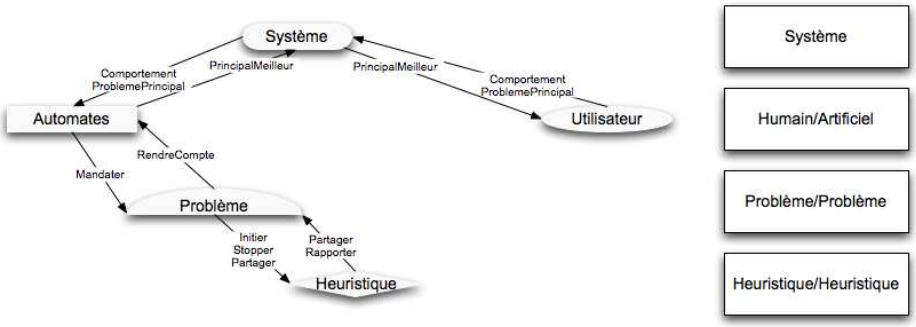

FIG. 3.12 – Ensemble des agents et types de messages.

Chapitre 4

Illustration

Introduction

Dans le chapitre précédent, nous avons exposé un système multi-agent de recherche heuristique permettant d'explorer trois synergies potentielles. L'objet de ce chapitre est de compléter cette description par un espace de recherche, ses générateurs ainsi que les testeurs sur lesquels nous avons expérimenté et de fournir les résultats concernant ces synergies.

Dans ce chapitre, nous allons décrire les éléments suivants nécessaires à une description complète de notre système :

Espace de recherche les réseaux de neurones

Voisinages ajout/suppression d'un neurone/lien ...

Agents heuristique algorithme génétique, recherche avec tabous, recherche à voisinage variable, recuit simulé

Testeurs évaluation d'une configuration de backgammon[1] et partage d'informations dans une grille d'agents

Nous avons choisi l'espace des réseaux de neurones car il est le principal format de fonctions soumis à des recherches heuristiques [Yao99]. Il s'agit donc d'un choix naturel pour étudier les synergies entre méthodes de recherche de fonctions[2]. Réseaux de neurones et émergence apparaissent souvent conjointement comme dans le titre d'un article fondateur de Hopfield [Hop82] : "Neural networks and physical systems with emergent collective computational abilities".

[1] Les règles du backgammon sont données en annexe page 155.

[2] Parmi les autres choix possibles, on peut citer les arbres de décision, les arbres de fonctions utilisés en programmation génétique.

Le choix de nos deux testeurs correspond à la recherche de deux exemples classiques dans le domaine de l'intelligence artificielle classique ou distribuée : la recherche d'une fonction d'évaluation pour un jeu comme le backgammon a donné lieu à de nombreux travaux dont certains ont abouti à des solutions de très haut niveau [Tes95] ; la possibilité de trouver une fonction déterminant le comportement d'une population d'agents cherchant à calculer une fonction collective est étudiée en système multi-agent et notamment en robotique [ØL03]. Les heuristiques que nous avons choisies correspondent à un ensemble classique.

Dans la suite du chapitre, nous décrivons l'espace des réseaux de neurones, ses générateurs, les heuristiques sous leur forme d'agents puis les deux testeurs servant à notre illustration. Ensuite nous abordons les résultats de nos expérimentations pour les trois synergies potentielles que nous avons décrites précédemment.

4.1 Espace de recherche des Réseaux de neurones

Nous donnons ci-dessous une description de la forme d'un réseau de neurones mais également de son interprétation calculatoire. Cette signification du point comme calcul est nécessaire à sa simulation. Nous donnons donc l'information nécessaire permettant de passer du réseau à la fonction qu'il calcule c'est-à-dire son comportement.

4.1.1 Forme

Un réseau de neurones est un tuple $< In, Out, Neurones, Liens >$ où

In Un ensemble d'entrées du réseau de neurones, chaque sommet d'entrée correspondant est muni d'un identifiant i.

Out Un ensemble de sorties du réseau ; les sommets de sorties ont une fonction d'agrégation agr_s et un identifiant s.

$Neurones$ un ensemble de neurones, chaque neurone est muni d'un identifiant n, d'une fonction d'agrégation agr_n, d'une fonction d'activation act_n et d'un biais b_n.

$Liens$ un ensemble de liens entre éléments des trois précédents ensembles. Chaque lien étant muni d'un couple (de,vers) et d'un poids $p(de, vers)$.

La terminologie des graphes nous permet de préciser cette définition : un réseau de neurones est un graphe orienté acyclique dont l'ensemble des sommets est $In \cup Out \cup Neurones$ et les arêtes les $Liens$.

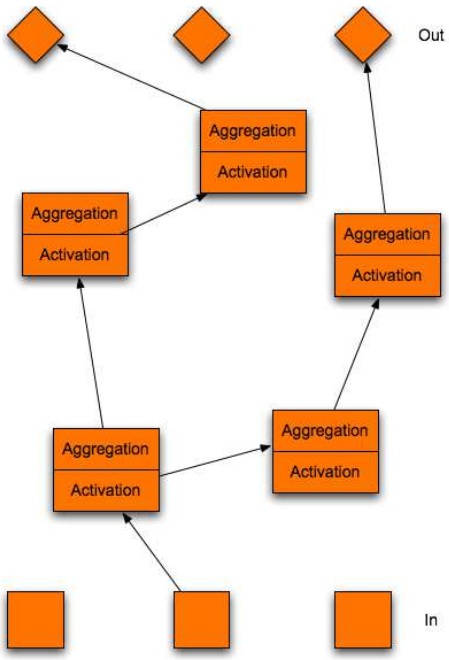

FIG. 4.1 – Représentation graphique d'un réseau de neurones

4.1.2 Sémantique

Une fois la syntaxe d'un réseau de neurones définie, on peut donner sa sémantique : les sommets d'entrée reçoivent une valeur qui se propage selon les liens vers les neurones suivants, chaque neurone ayant reçu l'ensemble de ses signaux les agrègent puis applique sa fonction d'activation au résultat ajouté à son biais puis propage cette valeur aux neurones suivants ... Une fois l'ensemble des signaux propagés, on peut alors lire la valeur des sommets de sorties qui nous donnent le résultat de l'activation.

Pour chaque sortie s, le calcul de cette sortie correspond à une valeur $f(s)$, f est calculée récursivement sur tous les sommets :

entrée pour un sommet d'entrée, la valeur est fournie qui fixe $f(i)$

neurone pour un neurone $f(n) = act_n(agr_n(\{p(e,n)*f(e)+b_n\}_{(e,n)\in Liens}))$

sortie pour un sommet de sortie, $f(s) = agr_s(\{p(e,s) * f(e)\}_{(e,s)\in Liens})$

4.1.3 Utilisation

Un simulateur fixe les valeurs des entrées d'un réseau et récupère les valeurs de ses sorties. Chaque activation du réseau fournit un ensemble d'entrées et un ensemble de sorties qui constituent un élément du comportement de ce réseau. L'interface entre le simulateur et le réseau se fait à travers les identifiants des sommets d'entrées et de sorties (cf. figure 4.2).

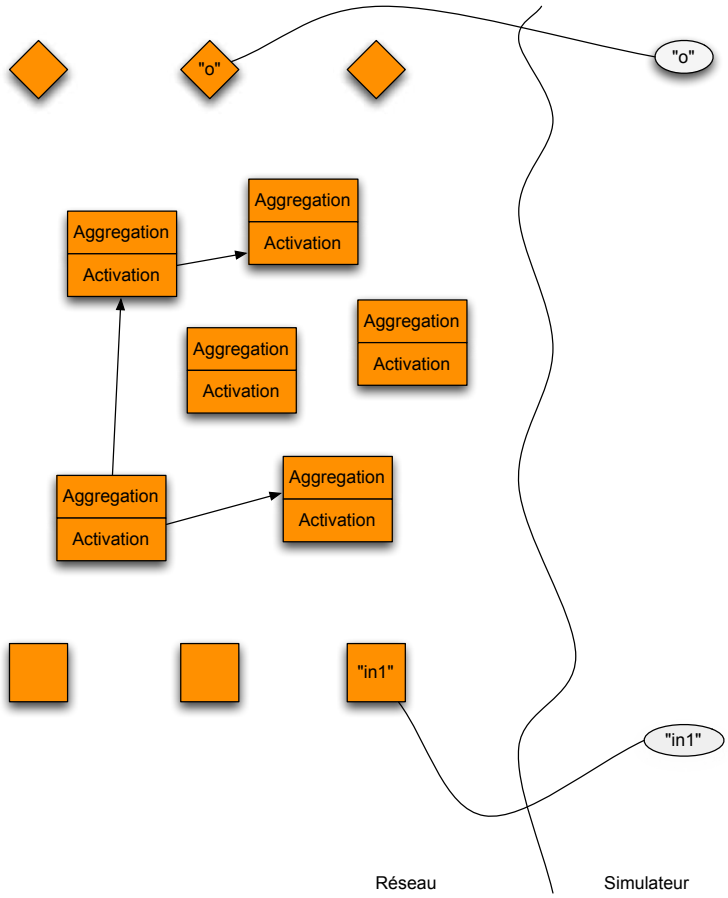

FIG. 4.2 – Un réseau de neurones et son interface reliés par les identifiants

4.1.4 Générateurs

Cette section contient la description informelle des générateurs que nous avons utilisés. Nous classons les générateurs selon qu'ils modifient des neurones, des liens ou la topologie du réseau.

Générateurs modifiant des neurones

Les générateurs modifiant les neurones conservent la topologie du réseau : ils ne changent pas le nombre de neurones ni les liens entre eux. Ils ne changent qu'une propriété d'un ou plusieurs neurones.

Changer le biais d'un neurone Pour un neurone particulier, on change la valeur de son biais.

Changer la fonction d'activation d'un neurone Pour un neurone, on change sa fonction d'activation. Les fonctions d'activation sont i) *identite* qui ne change rien ii) *signe* qui vaut 0 pour toute valeur négative, 1 sinon iii) sigmoïde qui à x associe $\frac{1}{1+e^{-x}}$

Changer la fonction d'agrégation d'un neurone Pour un neurone, on change la fonction d'agrégation. Nous disposons de deux fonctions d'agrégation : la somme des signaux entrants ou le produit des signaux entrants.

Générateurs modifiant des liens

Les générateurs modifiant les liens permettent de modifier les valeurs des poids sur les liens qui relient les neurones. L'application d'un de ces générateurs sur un réseau ne modifie pas sa topologie.

Multiplier le poids d'un lien par une constante Il s'agit de générer des voisins qui sont des copies de l'original dont un lien a vu son poids multiplié par une constante.

Additionner le poids d'un lien à une constante Les points de ce voisinage sont des identiques à l'original si ce n'est que pour chacun, un des liens est augmenté d'une constante.

Appliquer une modification selon une distribution normale Ce générateur consiste à appliquer une modification fondée sur une distribution normale (ou gaussienne) de moyenne nulle et de variance unitaire[3]. La modification se fait alors sur tous les poids du réseau.

[3] Ce générateur nécessite une source de hasard.

Générateurs modifiant la topologie

Les générateurs modifiant la topologie correspondent à l'ajout/suppression et non à la modification d'une entité du réseau.

Supprimer un neurone Il s'agit ici de supprimer un neurone dans l'ensemble *Neurones* ainsi que tous les liens dans *Liens* partant ou arrivant de ce neurone. Un voisin est engendré pour chaque neurone présent dans l'original.

Ajouter un neurone lié Il s'agit d'ajouter un neurone et deux liens (un entrant et un sortant) et ce sans créer de cycle dans le graphe.

Ajouter un lien entre noeuds Il s'agit ici de créer un nouveau lien dans le réseau. Seuls les ajouts ne créant pas de cycles dans le réseau sont considérés.

Supprimer un lien entre noeuds Ce générateur engendre un voisin par lien et correspond au réseau d'origine que l'on a privé de ce lien.

Relier entrées/sorties Il s'agit de rajouter tous les liens manquants sur $In \times Out$ en choisissant une valeur par défaut pour le poids du lien.

4.2 Agents heuristiques

Les agents de recherche heuristique sont chargés d'engendrer une trajectoire de recherche dans l'espace des réseaux de neurones. Dans cette section nous donnons un certain nombre de classes d'agents implémentant l'interface d'agent heuristique :

Interface Heuristique

Entrées

Initier : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩

Stopper : ⟨ ⟩

Boucler : ⟨ ⟩

Partager : ⟨ point :Point,valeur :Valeur ⟩

Sorties

Rapporter : ⟨ [⟨ point :Point,valeur :Valeur ⟩] ⟩

Boucler : ⟨ ⟩

Partager : ⟨ point :Point,valeur :Valeur ⟩

Pour toutes les classes que nous décrirons :

- *alea* désigne une source de hasard.
- Un message Rapport est envoyé dès que la condition d'arrêt est remplie et l'agent arrête sa recherche[4].
- Un message **Partager** : ⟨ point :Point,valeur :Valeur ⟩ est envoyé implicitement dès que l'agent ajoute à sa trajectoire un point constituant un progrès.
- La réception d'un message **Initier** : ⟨ testeur :Testeur,generateurs :[Generateur],depart :Point ⟩ initialise la mémoire de l'agent avec la définition du problème et envoie un message Boucler à l'agent heuristique lui-même.
- La réception d'un message **Stopper** : ⟨ ⟩ entraîne l'émission d'un rapport à l'agent problème contenant le meilleur point de la trajectoire.

Les deux points principaux à traiter pour définir une heuristique sont maintenant :

- Le traitement d'un message **Boucler** : ⟨ ⟩ qui correspond à un pas de recherche pour une heuristique itérative.
- Le traitement d'un message **Partager** : ⟨ point :Point,valeur :Valeur ⟩ qui définit comment l'agent intègre une solution partagée.

Chaque agent construira sa trajectoire au fil des pas de recherche. La trajectoire est la séquence des meilleurs couples ⟨ point,valeur ⟩ rencontrée au cours de la recherche. Pour chaque heuristique, nous donnons une idée de la manière dont elle construit cette trajectoire ainsi que la description de l'agent lui correspondant.

4.2.1 Algorithme génétique

Intuition La théorie de l'évolution de Charles Darwin contient l'idée que seuls les individus adaptés survivent, se reproduisent. Au cours de l'évolution, les individus tendent à être de plus en plus adaptés à leur environnement. Les algorithmes génétiques reprennent les grands traits de cette description : on considère une population de solutions qui sont sélectionnées selon leur degré d'adaptation, puis les solutions conservées donnent "naissance" à de nouvelles solutions tendant ainsi vers une adaptation accrue.

Le nom d'algorithme génétique couvre une classe d'algorithmes introduites par Holland [Hol75] et Goldberg [Gol89] inspirés de la théorie de

[4] Il ne s'envoie pas de message Boucler.

l'évolution. Ce type d'algorithme est habituellement décrit en trois phases qui font évoluer une population :

Évaluation Les individus de la population sont évalués par une fonction déterminant leur degré d'adaptation.

Sélection Les individus les plus adaptés survivent, les autres meurent.

Génération De nouveaux individus sont créés à partir des individus de la population.

L'expression de ces trois étapes se fait naturellement dans notre modèle de recherche heuristique en considérant qu'un individu est un point, que le testeur permet d'évaluer l'adaptation et que les générateurs permettent d'engendrer de nouveaux individus.

Notre algorithme génétique maintient une population de taille fixe, la sélection est faite par la conservation de la meilleure frange de la population et l'arrêt de la recherche se fait quand un nombre donné d'itérations n'a pas donné lieu à amélioration[5].

Mémoire Cet agent heuristique doit maintenir l'information de sa population courante. Il faut que les paramètres de taille, la fraction[6] de la population conservée et le nombre d'itérations d'arrêt fassent partie de sa mémoire.

Ces éléments en plus de ceux que doivent mémoriser toutes les classes d'heuristiques nous donnent le type de mémoire suivant : ⟨ population :[Point], parametres : ⟨ taille :Entier,iterations :Entier,fraction :Entier ⟩ , trajectoire :[⟨ Point,Valeur ⟩], probleme : ⟨ testeur :Testeur, generateurs :[Generateur] ⟩ ⟩

Pour les classes d'agent suivantes, nous omettrons de mentionner *trajectoire* et *probleme* puisqu'ils sont partagés par toutes les classes. Nous appellerons cette classe d'agent *Genetic*.

Fonctions de traitement Nous devons maintenant donner les fonctions de traitement de chaque type de messages contenus dans l'interface d'entrée. Le message Boucler va servir à effectuer un pas de recherche, le message Partager sert à intégrer un point de l'extérieur.

La réception d'un message Boucler correspond au déclenchement d'un pas de la recherche.

[5] La valeur la plus élevée dans la population n'a pas changé.

[6] Exprimée par un pourcentage.

Algorithme 8 : Traitement par un agent Genetic d'un message Boucler

```
Data : message, un message de type Boucler
Data : memoire, la memoire courante
1  begin
       /* Pas de recherche                                          */
2      choisis = {};
       /* Sélection                                                 */
3      while taille(choisis) ≤
       memoire.fraction*100/taille(memoire.population) do
4          un = extrait(memoire.population,alea);
5          deux = extrait(memoire.population,alea);
6          choisi = meilleur([un,deux]);
7          choisis = union(choisis,{choisi});
8      end
       /* Génération                                                */
9      generes = {};
10     for individu ∈ choisis do
11         generateur = extrait(memoire.probleme.generateurs), alea);
12         genere = extrait( generateur(individu) , alea);
13         generes = union(generes,{genere});
14     end
       /* Evaluation                                                */
15     evalues = {};
16     testeur = memoire.probleme.testeur;
17     forall point ∈ generes do
18         evalues = union(evalues, { ⟨ point,testeur(point) ⟩ });
19     end
       /* Construire la nouvelle mémoire                            */
20     memoire = memoire    avec    population=union(evalues,
       meilleurs(population,n));
       /* On continue la recherche si le critère d'arrêt n'est
          pas satisfait                                             */
21     if !arret then
22         boucler = Boucler();
23         return ⟨ memoire,[boucler] ⟩;
24     end
25     else
26         rapport = Rapporter(pere,meilleur(memoire.trajectoire));
27         return ⟨ memoire,[rapport] ⟩;
28     end
29 end
```

La réception d'un message Partager (cf. algorithme 9) contient un point à intégrer dans la population de l'agent. Ce point est simplement ajouté à la population de l'agent.

Algorithme 9 : Traitement par un agent Genetic d'un message Partager

Data : message, un message de type Partager
Data : memoire, la memoire courante
1 **begin**
  `/* Ajout à la population */`
2 memoire = memoire avec population=union(population,point(message));
3 **return** ⟨ *memoire,[]* ⟩;
4 **end**

4.2.2 Recuit simulé

Intuition L'idée de recuit provient du domaine de la métallurgie. Il s'agit d'alterner des périodes de cuisson et de refroidissement lent afin d'augmenter la stabilité d'un objet en métal. Les périodes chaudes permettent des mouvements dans le métal, puis durant les périodes de refroidissement, les constituants du métal peuvent s'arranger de la manière la plus stable possible.

La méthode du recuit simulé a été introduite par Kirkpatrick Gelatt et Vecchi [KGV83]. Elle consiste à conserver une pseudo-température guidant la recherche de la manière suivante : plus il fait chaud, plus la recherche est aléatoire, plus il fait froid, plus la recherche est une descente de gradient.

Dans sa présentation habituelle, l'algorithme nécessite la précision de plusieurs éléments :

probabilité de transition Le passage d'un point à un autre selon les valeurs des points et la température. Si la valeur du nouveau point $v_{nouveau}$ est supérieure ou égale à celle du point courant $v_{courant}$, on choisit le nouveau point. Sinon, pour une température t, on choisit le nouveau point avec une probabilité $e^{\frac{v_{nouveau}-v_{courant}}{t}}$.

un voisinage Une fonction permettant d'obtenir un point voisin.

un échéancier de température Il s'agit de la température initial et d'un facteur strictement inférieur à 1 par lequel on multiplie la température à chaque pas de recherche.

Mémoire Dans le cadre d'un recuit simulé, il est nécessaire de maintenir les paramètres de l'algorithme : la température (la température initiale est 1.0), le facteur faisant décroître cette température et le point courant qui est potentiellement différent du meilleur point de la trajectoire.

Fonctions de traitement La réception d'un message Partager (cf. algorithme 10) est traité en remplaçant le point courant par le point partagé si celui-ci est de meilleure qualité ou que la température est élevée.

Algorithme 10 : Traitement par un agent Recuit d'un message Partager

```
  Data : message, un message de type Partager
  Data : memoire, la memoire courante
1 begin
    /* Si le partage est meilleur, il remplace le point
       courant                                          */
2   if message.valeur > memoire.courant.valeur ou alea <
    memoire.temperature then
3       return ⟨ (memoire avec courant= ⟨
        message.point,message.valeur ⟩ ),[] ⟩;
4   end
5   else
6       return ⟨ memoire,[] ⟩;
7   end
8 end
```

La réception d'un message Boucler (cf. algorithme 11) donne lieu à l'exécution d'un pas de recherche. L'agent produit un nouveau point puis décide de s'y déplacer ou non selon les valeurs de ce point et du point courant et de sa température. Finalement, il diminue cette température en la multipliant par le $facteur$ contenu dans sa mémoire.

Algorithme 11 : Traitement par un agent Recuit d'un message Boucler

```
Data : message, un message de type Boucler
Data : memoire, la memoire courante
begin
    /* Prendre un des générateurs                                  */
    generateur = extrait(memoire.probleme.generateurs), alea);
    /* Choisir un point                                            */
    nouveau = extrait(generateur(courant),alea);
    valeur = testeur(probleme(memoire))(nouveau);
    /* Si on a un progrès ou une tirage aléatoire favorable
       */
    if valeur > valeur(courant) ou alea < temperature(memoire) then
        memoire = memoire    avec    courant = ⟨ courant,valeur ⟩ ;
    end
    /* Diminuer temperature                                        */
    memoire = memoire    avec    temperature =
    (temperature*facteur);
    if !arret then
        boucler = Boucler();
        return ⟨ memoire,[boucler] ⟩;
    end
    else
        /* Ici le meilleur n'est pas forcément le point
           courant                                                 */
        rapport = Rapporter(pere,meilleur(trajectoire(memoire)));
        return ⟨ memoire,[rapport] ⟩;
    end
end
```

4.2.3 Recherche tabou

Intuition La recherche dans un grand espace peut devenir très longue mais surtout tomber dans des zones où la meilleure solution dans les alentours est la solution actuelle. Dans ces cas, on cherche un moyen d'en sortir. L'idée d'une recherche avec tabous est de forcer le mouvement en empêchant de revenir là d'où on vient en prenant de nouvelles bonnes solutions même si elles ne sont pas meilleures.

L'heuristique *Tabou* introduite par Glover [Glo89] consiste à éviter de revenir sur des solutions déjà envisagées en les marquant d'un tabou et ce tout en se déplaçant toujours vers un bon point. Cette idée pose quelques problèmes de mise en oeuvre :

- L'espace à explorer est souvent très grand et la liste des points à marquer peut devenir énorme. Un paramètre de taille de la liste est alors nécessaire.
- Pour savoir si on est revenu à une solution déjà vue, il faut se doter d'une relation d'égalité. Cette relation n'est pas toujours facile à obtenir, notamment quand la définition d'une solution inclut des nombres flottants ou des graphes, ce qui est le cas des réseaux de neurones.

Mémoire La mémoire d'un agent de recherche Tabou doit inclure le point courant (*courant*) une liste de points tabous (*tabous*) ainsi que la taille (*taille*) limite de la liste que l'on conservera.

Fonctions de traitement Le traitement d'un message Partager par un agent Tabou (cf. algorithme 12) doit forcement vérifier que l'élément partagé n'est pas un membre de la liste de tabous. On teste également si cet élément est de bonne qualité sans quoi il n'est pas retenu.

Le traitement d'un message Boucler (cf. algorithme 13) correspond au passage au meilleur élément des voisins privé des éléments de la liste de tabous. L'agent choisit alors un générateur au hasard et prend le meilleur élément du voisinage qui ne soit pas tabou.

Algorithme 12 : Traitement par un agent Tabou d'un message Partager

Data : message, un message de type Partager
Data : memoire, la memoire courante
1 **begin**
2 **if** *!contient(memoire.tabous,message.point) et message.valeur > courant.valeur* **then**
3 memoire = memoire avec courant = ⟨ message.point,message.valeur ⟩ ;
`/* L'ajout de ce point aux tabous */`
4 memoire = memoire avec tabous = ajoutdroite(tabous,message.point,memoire.taille);
5 **end**
6 **return** ⟨ *memoire,[]* ⟩;
7 **end**

4.2.4 Changement de voisinage

Intuition Les agents heuristiques se déplacent de proche en proche en s'orientant selon ce qu'un testeur leur dit de leur progrès. L'idée derrière une recherche à voisinage variable est de changer ce qu'on entend par "de proche en proche" au fil de la recherche. Cela consiste en un changement du voisinage quand la recherche d'une meilleure solution est bloquée sur le voisinage courant.

La descente selon un voisinage variable est une technique introduite par Mladenovic et Hansen [HM97]. En partant du constat que l'heuristique de descente de gradient tombe souvent sur un minimum local, on propose de changer de générateur et donc de voisinage dès que le voisinage courant ne permet plus de progresser. L'idée est qu'un minimum local n'est local que selon la définition du voisinage associé.

Dans notre modèle de recherche, cette heuristique s'exprime naturellement par une itération sur les générateurs. On cherche alors à progresser jusqu'à ce qu'il n'y ait plus de progression, dans ce cas on change de voisinage afin de continuer la progression.

Nous définissons deux variantes : la première *Exhaustif* qui à chaque itération prendra le meilleur des éléments du voisinage envisagé et devra

Algorithme 13 : Traitement par un agent Tabou d'un message Boucler

```
Data : message, un message de type Boucler
Data : memoire, la memoire courante
1  begin
       /* Prendre un des générateurs                          */
2      generateur = extrait(memoire.probleme.generateurs), alea);
3      produits = generateur(courant(memoire));
       /* Eliminer les points tabous                          */
4      produits = difference(produits,tabous(memoire));
       /* Evaluer les points produits                         */
5      evalues = ;
6      test = testeur(memoire);
7      forall produit ∈ produits do
8          evalues = union(evalues, ⟨ produit,test(produit) ⟩ );
9      end
       /* Prendre le meilleur qui ne soit pas tabou.          */
10     nouveau = meilleur(evalues);
11     memoire = memoire   avec   courant = meilleur;
12     memoire = memoire   avec   tabous =
       ajoutdroite(tabous,meilleur,memoire.taille);
13 end
14 if !arret then
15     boucler = Boucler();
16     return ⟨ memoire,[boucler] ⟩;
17 end
18 else
       /* Ici le meilleur n'est pas forcément le point courant
          */
19     rapport = Rapporter(pere,meilleur(memoire.trajectoire));
20     return ⟨ memoire,[rapport] ⟩;
21 end
```

donc évaluer tous les éléments du voisinage[7] et une seconde *Direct* qui dès qu'un voisin est meilleur que la solution courante le sélectionne et passe à l'itération suivante[8].

Mémoire La mémoire d'un agent pratiquant un changement de voisinage doit retenir le voisinage (générateur) courant *gcourant* ainsi que le point courant sur lequel il fait sa recherche *pcourant*.

Fonctions de traitement Le traitement d'un message Partager est le même pour les deux variantes : l'agent remplace son point courant par le point reçu si celui-ci constitue un progrès dans la recherche de l'agent.

Algorithme 14 : Traitement par un agent Variable d'un message Partager

```
  Data : message, un message de type Partager
  Data : memoire, la memoire courante
1 begin
    /* Si le point est meilleur que le point courant.    */
2   if message.valeur > memoire.pointcourant.valeur then
3       memoire = memoire   avec   pcourant = ⟨
        message.point,message.valeur ⟩ ;
4   end
5   return ⟨ memoire,[] ⟩;
6 end
```

Le traitement d'un message de type Boucler (cf. algorithme 15) effectue une recherche d'un meilleur voisin selon le générateur courant. Si celui-là existe, on passe au pas de recherche suivant, sinon on change de voisinage avant de boucler. Pour un agent de classe Direct (cf. algorithme 16), dès qu'un membre du voisinage constitue un progrès il devient le point courant, le pas de recherche se termine.

4.3 Testeurs

Un testeur sert à calculer la valeur que l'on peut attribuer à un point de notre espace de recherche. Le testeur sert à modéliser le problème pour

[7] Ceci correspond à l'idée de l'algorithme *Variable Neighborhood Search*.
[8] Ceci correspond à l'algorithme *Variable Neighborhood Descent*.

Algorithme 15 : Traitement par un agent Exhaustif d'un message Boucler

```
Data : message, un message de type Boucler
Data : memoire, la memoire courante
begin
    evalues = {};
    testeur = memoire.probleme.testeur;
    for produit ∈ memoire.gcourant(memoire.pcourant) do
        evalues = union(evalues, ⟨ produit, testeur(produit) ⟩ );
    end
    candidat = meilleur(evalues);
    if candidat.valeur > memoire.pcourant.valeur then
        /* On se déplace en gardant le même générateur    */
        memoire = memoire    avec    pcourant = candidat;
    end
    else
        /* Si on ne progresse pas, on change de voisinage */
        memoire = memoire    avec    gcourant =
        succ(memoire.probleme.generateurs,gcourant);
    end
    if !arret then
        boucler = Boucler();
        return ⟨ memoire,[boucler] ⟩;
    end
    else
        /* Ici le meilleur est toujours le point courant  */
        rapport = Rapporter(pere,memoire.pcourant);
        return ⟨ memoire,[rapport] ⟩;
    end
end
```

Algorithme 16 : Traitement par un agent Direct d'un message Boucler

```
   Data : message, un message de type Boucler
   Data : memoire, la memoire courante
 1 begin
 2     generes = memoire.gcourant(memoire.pcourant);
 3     testeur = memoire.probleme.testeur;
 4     point = premier(generes);
       /* On initialise le candidat avec le premier point   */
 5     candidat = ⟨ point, testeur(point) ⟩ ;
       /* On parcourt jusqu'à trouver le premier point
          meilleur que le point courant                     */
 6     repeat
 7         point = succ(generes,point);
 8         valeur = testeur(point);
 9         if valeur > candidat.valeur then
10             candidat = ⟨ point,valeur ⟩ ;
11         end
12     until candidat.valeur > memoire.pcourant.valeur ou point ==
       dernier(generes) ;
13     if candidat.valeur > memoire.pcourant.valeur then
           /* On se déplace en gardant le même générateur    */
14         memoire = memoire    avec    pcourant = candidat;
15     end
16     else
17         memoire = memoire    avec    gcourant =
           succ(memoire.probleme.generateurs,gcourant);
18     end
19     if !arret then
20         boucler = Boucler();
21         return ⟨ memoire,[boucler] ⟩;
22     end
23     else
           /* Ici le meilleur est le point courant           */
24         rapport = Rapporter(pere,pcourant);
25         return ⟨ memoire,[rapport] ⟩;
26     end
27 end
```

lequel on cherche des solutions. Nous donnons ici les deux exemples que nous avons utilisés dans nos expériences.

4.3.1 Testeur du jeu de backgammon

Notre premier testeur correspond à la recherche d'une bonne fonction de notation pour le jeu de backgammon. Nous cherchons une fonction qui à une configuration de backgammon associe une note[9].

Cette fonction de notation est alors utilisée dans le cadre d'une méthode exploratoire utilisée par un joueur automatique :

1. Déterminer les coups possibles.
2. Évaluer les configurations accessibles.
3. Jouer le coup permettant d'atteindre la configuration la mieux notée.

Dans le cas où plusieurs configurations ont la même note, un ordre total préalable est utilisé pour les départager.

Interface entre configuration et réseau de neurones

Les règles du backgammon sont données en annexe. Nous décrivons ici comment une configuration de ce jeu est interfacée avec un réseau de neurones afin que celui-ci puisse lui attribuer une note.

Le backgammon se joue sur une table de jeu découpée en 4 quadrants ou *jans* , constitués chacun de 6 cases ou *flèches* . Une position supplémentaire est la barre où on trouve les pions captifs.

Afin de se représenter graphiquement l'interface correspondante, nous déplions le plateau pour en faire une présentation en ligne (cf. figure 4.4).

Nous associons à chacune des flèches du plateau un noeud qui correspondra au nombre de pions présents sur cette position. Les autres informations fournies au réseau sont : le joueur dont c'est le tour, le nombre de coups déjà joués ou encore une estimation de distance au gain pour chaque joueur[10] (cf. figure 4.5). L'ensemble sert d'entrée au réseau de neurones qui présente une seule sortie correspondant à la note attribuée.

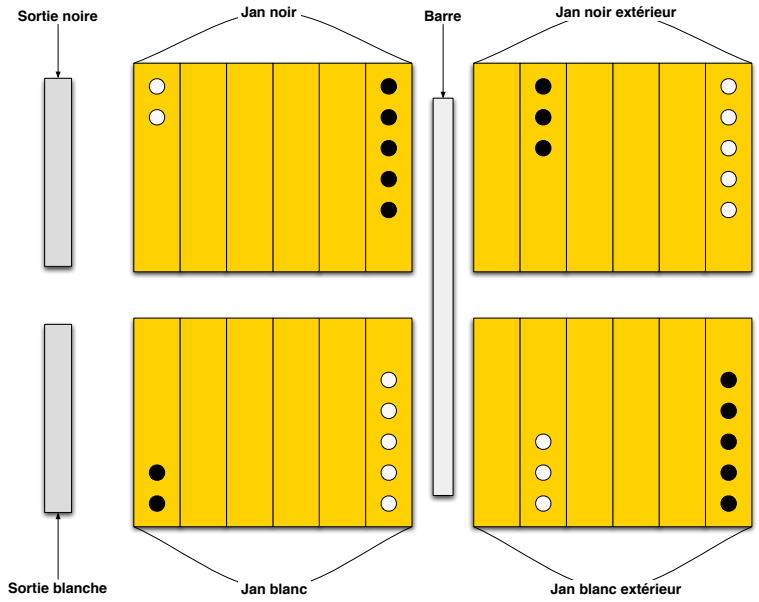

FIG. 4.3 – La table de jeu du backgammon

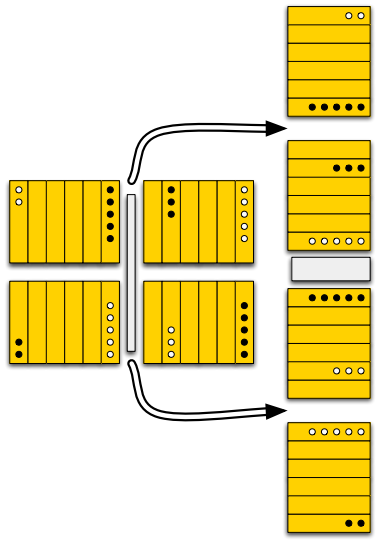

FIG. 4.4 – Vue habituelle et vue en ligne

[9]Nous n'utilisons pas ici le terme de valeur, nous appelons valeur ce que le simulateur associe au réseau et non ce que ce réseau calcule.

[10]Cet estimateur est la somme sur l'ensemble des pions d'un joueur de la distance entre

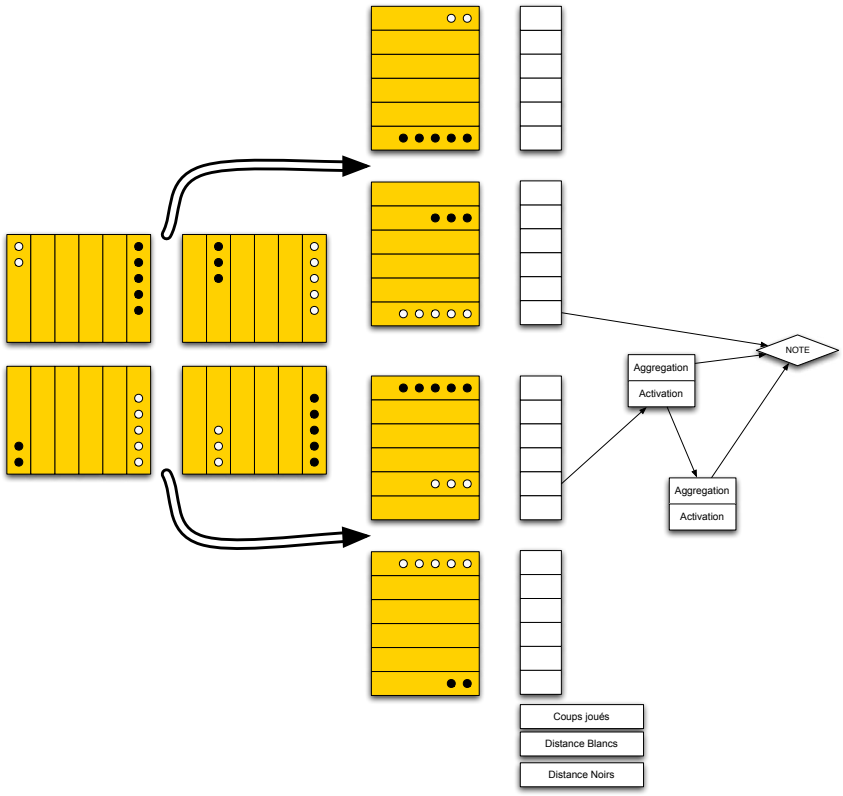

FIG. 4.5 – L'interface entre configuration et réseau calculant sa note

Scénario

Dans le cadre de la simulation d'une partie de backgammon contre un joueur automatique, on simule plusieurs partie entre le réseau et une fonction de référence, tous deux utilisés comme fonction de notation. Les lancers de dés de chaque partie sont déterminés de manière pseudo-aléatoire : le scénario d'une simulation consiste en une séquence de parties pour chacune desquelles nous donnons la graine servant à initialiser un générateur pseudo-aléatoire. Pour chaque partie, un nombre de coups maximum fait également partie du scénario afin de borner le temps de simulation d'une partie. La

ce pion et l'objectif.

valeur utilisée dans toutes nos expériences est 100 et dépasse le nombre maximal de coups joués dans la plupart des parties entre humains.

Fonctions d'évaluation

La fonction d'évaluation dans le cas du backgammon doit aller dans le sens de "plus un joueur gagne mieux il est évalué". On note le nombre de parties gagnées par le point g le nombre de parties perdues p et le nombre de parties nulles n de ce résultat.

$$\frac{g-p}{n+1} \tag{4.1}$$

L'addition de 1 est un simple artifice pour éviter toute division par 0. Cette fonction favorise les gains tout en évitant de récompenser les solutions aboutissant à des parties nulles.

Comportement fourni par l'utilisateur

Le comportement fourni par l'utilisateur est fourni sous la forme de couples de configurations : dans chaque couple, le premier élément contient la configuration présentée au joueur humain et le second la configuration une fois son coup joué. Dans le cadre de cette thèse, le comportement utilisé correspond à l'enregistrement des coups joués par l'auteur sur le scénario des expérimentations contre la fonction de référence[11].

4.3.2 Testeur du partage d'information dans une grille

Cette section décrit notre second exemple d'illustration de testeur. Il s'agit maintenant de chercher un réseau de neurones qui une fois déployé dans une grille permette d'y calculer un gradient.

Le cadre de ce simulateur est une grille en deux dimensions peuplée de cellules. Dans cette grille, chaque cellule peut percevoir l'état de ses voisins.

Chaque agent situé sur une case de coordonnées i, j perçoit une valeur entière de l'environnement pour cette case $e_{i,j}(t)$. L'objectif local de chaque cellule est de calculer sa valeur cellule $c_{i,j}(t)$. Chaque agent perçoit également les valeurs cellules de ses voisins.

L'exécution de la grille se fait par tour en temps discret. A chaque tour, toutes les cellules calculent leur $c_{i,j}(t+1)$ en fonction de $e_{i,j}(t)$ et

[11] Au cours de cet enregistrement, toutes les rencontres ont été gagnées par l'humain.

des $\{c_{k,l}\}_{i-1\leq k\leq i+1, j-1\leq l\leq j+1}$[12].

L'objectif global est de calculer une fonction de l'ensemble des valeurs de l'environnement. On voit donc que nous cherchons à calculer de manière distribuée une fonction qu'aucun des agents ne peut calculer en fonction de sa perception.

Interface avec un réseau de neurones

L'interface entre le simulateur et notre point se fait à travers un neurone pour chacune des valeurs définies auparavant : $e_{i,j}(t)$ et des $\{c_{k,l}\}_{i-1\leq k\leq i+1, j-1\leq l\leq j+1}$. La seule sortie du réseau correspond à la valeur de l'agent au temps suivant soit $c_{i,j}(t+1)$

Scénario

Un scénario dans le cas du partage d'information dans une grille doit permettre de décrire tous les événements prenant place dans la grille ainsi que sa configuration générale. Il s'agit de spécifier la taille de cette grille mais également les changements de valeur de l'environnement pour chacune des cases de cette grille. Un scénario est donc formé de la taille de la grille ainsi que d'une séquence de pas de temps pour chacun desquels des changements des $e_{i,j}$ sont spécifiés.

Solution fournie pour le problème du gradient

La solution que nous fournissons en intension pour le problème de gradient est la suivante :

$a_{i,j}(t+1) = max(e_{i,j}(t), max(\{a_{k,l}\}_{i-1\leq k\leq i+1, j-1\leq l\leq j+1}) - 1)$.

Il s'agit pour l'agent de prendre le maximum entre sa valeur d'environnement et le maximum des valeurs agent de ses voisins moins 1. Pour comprendre le fonctionnement de cette solution, prenons le cas simple où un seul agent a une valeur d'environnement non nulle et tous les agents sont initialisés avec $a_{i,j} = 0$. Au premier pas, cet agent garde sa valeur puisque le maximum des valeurs de ses voisins est nul mais ses voisins prennent sa valeur agent décrémentée. Au fil des pas de temps, les voisins des agents venant juste de changer leur valeur vont eux-mêmes prendre cette valeur décrémentée pour aboutir à un gradient (cf. figure 4.6).

[12] Les opérations sur les indices de cases se font modulo la taille de la grille, aussi l'indice de la dernière case plus un pointe la première case, ce qui constitue une vision "torique". Il n'y a donc pas d'agents au bord de la grille.

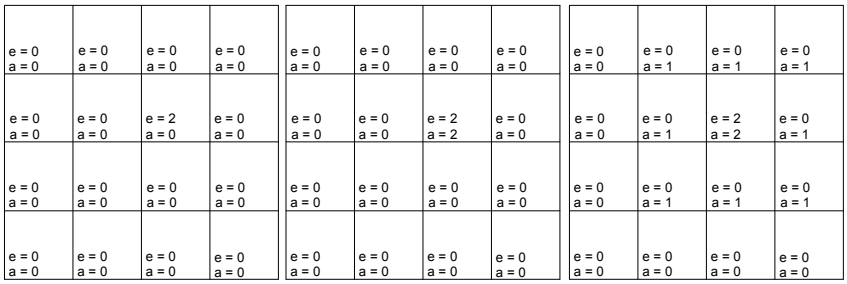

FIG. 4.6 – Diffusion d'un gradient par la solution fournie.

Fonction d'évaluation

Pour chaque pas de temps constituant la simulation, pour chaque cellule on calcule l'écart entre sa valeur cellule et celle qu'elle devrait avoir dans le gradient pour donner un $\delta_{i,j,t}$. Ces valeurs sont additionnées sur l'ensemble de la grille et sur l'ensemble de la simulation pour donner Δ. La valeur associée au réseau de neurones est $-\Delta$.

4.4 Implémentation

Nous donnons quelques informations sur l'implémentation effectuée dans la thèse :

- Les agents sont codés en Java et utilise la plate-forme de distribution de messages CoRe DMS.
- Les expériences sont décrites dans deux langages de programmation : i) Java sert à décrire les agents, les types de message et permet de bénéficier de nombreuses vérifications statiques ii) Python à travers l'interpréteur jython [Jyt] qui permet de décrire les paramètres de chaque expérience de manière plus riche qu'un fichier de propriétés sans avoir à créer un exécutable par expérience.
- L'ensemble de l'implémentation correspond à la traduction en Java de l'ensemble des fonctions de traitements que nous avons détaillées jusqu'ici plus la mise en place d'un agent générique implémentant notre modèle fonctionnel.

4.5 Synergie d'heuristiques

Jusqu'ici nous avons défini des synergies potentielles, les éléments que nous venons de décrire nous permettent maintenant de déterminer si ces synergies peuvent être observées.

Nous commençons l'exposé des résultats de nos expérimentations. Pour l'ensemble de ces résultats, ce qui nous intéresse est l'impact collectif : nous donnons toujours le gain de valeur obtenu par le système au cours de sa recherche soit entre le premier et le dernier point de sa trajectoire.

Cette section contient les résultats d'expériences où un seul problème a été envisagé en faisant varier la population d'agents heuristiques cherchant une solution. L'intérêt de cette première partie des résultats expérimentaux est de voir si notre agentification d'algorithmes heuristiques et en particulier la possibilité d'échanger des solutions en cours de recherche permet d'obtenir un gain sur plusieurs recherches parallèles sans interaction. Dans la suite de ce chapitre, nous distinguerons ces deux situations :

concurrence les agents communiquent leurs progrès en cours de recherche

parallélisme les agents mènent leurs recherches séparément[13]

Cette section traite des deux exemples d'illustration. Pour chacun, nous comparons chaque heuristique prise séparément ainsi que des populations d'heuristiques homogènes et hétérogènes.

4.5.1 Description de la méthodologie des expériences

Les expériences sont menées à l'aide du système multi-agent décrit auparavant. Toutes les recherches sont initiées à partir d'un même réseau de neurones constitué de l'ensemble des entrées et sorties du problème, d'un seul neurone relié à toutes les entrées et sorties par des liens de poids nul.

Pour chaque expérience, la population d'agents de recherche est spécifiée par le nombre d'agents de chaque classe. Chaque population est déclinée en versions parallèle et concurrente. Les paramètres du critère d'arrêt changent selon le testeur pour des raisons de temps de calcul : le testeur pour le backgammon prend plus de temps que celui de la grille ce qui nous a forcé à réduire le nombre d'évaluations pour mener à bien nos expérimentations.

[13]Nous utilisons les termes de parallélisme et de concurrence pour distinguer le cas où les agents partagent leurs progrès de celui où ils ne le font pas. Il ne s'agit pas de concurrence au sens d'une compétition, les agents sont concurrents car ils s'exécutent ensemble : simultanément et en échangeant des messages. Des agents parallèles s'exécutent simultanément mais sans cet échange.

Chaque expérience est répétée 100 fois afin d'obtenir des mesures significatives.

4.5.2 Backgammon

Le problème pour le jeu de backgammon est le suivant : gagner le plus de parties possibles parmi 7 contre un adversaire jouant avec la fonction d'évaluation suivante :

- Plus le nombre de sauts à effectuer par son adversaire pour finir est grand, meilleure est la position. Ceci rend le joueur glouton puisque pour augmenter la distance de l'autre il faut capturer ses pions.
- Moins il y a de positions où l'adversaire peut lui prendre un pion, meilleure est la situation. Ceci rend le joueur prudent puisqu'il cherche à ne pas exposer ses pions.

Le critère d'arrêt est le suivant : un agent heuristique ne peut pas dépasser un total de 100 appels au testeur, il ne peut pas non plus dépasser 50 appels sans avoir progressé.

Heuristiques isolées

La première comparaison est celle entre des populations constituées d'un seul agent. Cette comparaison permet d'évaluer chaque type d'agent individuellement avant de l'évaluer en collectivité. Le résultat correspond au gain produit par la recherche pris en moyenne sur les 100 exécutions effectuées pour chaque population.

classe	moyenne (écart type)
Genetic	1.249 (0.979)
Recuit	1.167 (0.877)
Tabou	0.708 (0.577)
Direct	0.730 (0.629)
Exhaustif	0.643 (0.409)

Deux classes d'heuristiques semblent se distinguer : les heuristiques Exhaustif Direct et Tabou d'une part et les heuristiques Genetic et Recuit d'autre part. Ce premier résultat est intéressant car il semble suggérer que parmi les différences existant entre les différentes heuristiques, la caractéristique la plus discriminante est la manière dont les points à explorer sont produits. En effet, Genetic et Recuit tirent des points au hasard dans l'ensemble des voisinages alors que les trois autres construisent un seul voisinage à chaque tour.

Ce critère semble plus discriminant que certaines hypothèses qui peuvent sembler plus naturelles comme la distinction habituelle entre heuristique à base de population (Genetic) et heuristique fondée sur un unique individu.

Populations homogènes

Les collectifs homogènes sont des populations de recherche qui sont tous de la même classe. Pour chaque heuristique, nous avons mené les expériences avec des populations de 1, 4 et 5 agents.

Recherches en parallèle Nous commençons par des populations homogènes en parallèle.

classe	1	4	5
Genetic	1.249 (0.979)	2.223 (1.668)	2.277 (1.603)
Recuit	1.167 (0.877)	2.250 (1.796)	2.347 (1.705)
Tabou	0.708 (0.577)	1.533 (1.739)	1.434 (1.359)
Direct	0.730 (0.629)	1.146 (0.730)	1.276 (0.982)
Exhaustif	0.643 (0.409)	1.046 (0.724)	1.068 (0.437)

Le gain de performance est normal puisque nous comparons les meilleurs points trouvés par une population de recherche : ainsi un nouvel agent ne peut qu'améliorer la meilleure solution.

La ligne de l'heuristique Tabou illustre un problème de notre système. En effet, notre système contient de nombreuses sources de non-déterminisme dont les effets peuvent être importants. Pour des valeurs proches comme dans le cas présent, le hasard peut provoquer ce type de résultat aberrant. Nous discutons en fin de chapitre de nos pistes pour régler ce type de problème.

Recherches en concurrence Nous donnons maintenant les résultats de nos expériences quand les interactions sont activées dans ces populations de recherche. Les agents sont maintenant tous informés de la découverte d'un progrès et intègrent ce progrès dans leur recherche.

classe	1	4	5
Genetic	1.249 (0.979)	2.189 (1.669)	2.095 (1.369)
Recuit	1.167 (0.877)	2.683 (2.020)	3.680 (2.448)
Tabou	0.708 (0.577)	1.760 (2.024)	1.663 (1.722)
Direct	0.730 (0.629)	1.576 (1.645)	1.465 (1.428)
Exhaustif	0.643 (0.409)	1.634 (1.603)	1.428 (1.370)

On observe une tendance générale qui montre encore une fois un gain à l'interaction. Ce gain signifie que nous observons encore une fois une syner-

gie entre agents de recherche dans le cas où ceux-ci cherchent de manière concurrente.

L'autre point intéressant est que la performance diminue pour certaines heuristiques quand on passe de 4 à 5 agents. Cela montre qu'il n'est pas toujours bon d'être nombreux si l'interaction est activée car cela peut provoquer le partage de solutions peu prometteuses qui handicapent la recherche ensuite.

Finalement, le nombre optimal d'agents change selon l'heuristique, cela pose la question du nombre optimal sur l'ensemble des entiers pour chaque heuristique et en particulier, on peut se demander si on gagne toujours à rajouter un agent de recuit simulé.

Populations hétérogènes

Les collectifs hétérogènes sont constitués d'agents faisant partie de classes différentes. A travers ces expériences, nous cherchons à savoir si il existe une relation entre la qualité d'une heuristique lorsqu'elle est utilisée seule et sa qualité dans un cadre collectif. Pour cela, pour chaque heuristique nous comparons sa qualité seule (Seul), la qualité de la population contenant un agent de chaque classe (Tous) et finalement la qualité de la population où nous avons retiré l'agent de la classe étudiée (Sans).

Comme nous étudions maintenant des populations d'agents, nous détaillons les résultats selon que les recherches sont parallèles ou concurrentes.

Recherches en parallèle Ces résultats correspondent aux populations sans communications.

classe	Seul	Sans	Tous
Genetic	1.249 (0.979)	1.482 (1.237)	2.114 (1.386)
Recuit	1.167 (0.877)	1.541 (1.203)	2.114 (1.386)
Direct	0.730 (0.629)	1.794 (1.411)	2.114 (1.386)
Tabou	0.708 (0.577)	1.756 (1.506)	2.114 (1.386)
Exhaustif	0.643 (0.409)	1.790 (1.483)	2.114 (1.386)

Ces résultats correspondent à l'intuition puisque la population privée d'un bon agent perd en qualité alors que la perte d'un mauvais agent dégrade moins la performance de la population.

Recherches en concurrence Les résultats en concurrence correspondent au cas où les agents envoient un message à chaque progrès réalisé. Les autres agents peuvent alors intégrer ces progrès au cours de leur recherche.

classe	Seul	Sans	Tous
Genetic	1.249 (0.979)	1.742 (1.606)	2.345 (1.925)
Recuit	1.167 (0.877)	1.521 (1.420)	2.345 (1.925)
Direct	0.730 (0.629)	2.103 (1.806)	2.345 (1.925)
Tabou	0.708 (0.577)	2.028 (1.602)	2.345 (1.925)
Exhaustif	0.643 (0.409)	2.074 (1.821)	2.345 (1.925)

Nous observons ici une tendance générale qui montre un gain à l'interaction dans ces populations de recherche hétérogènes. Il s'agit ici d'une synergie de recherche observée : nous avons des résultats qui montrent qu'un gain de performance est obtenu dans un système qui ne diffère du système de référence que par des interactions supplémentaires. On peut alors affirmer que nous avons ici un tout supérieur à la somme de ses parties.

Par ailleurs, nous observons que l'introduction de notre interaction ne semble pas changer l'ordre entre les heuristiques ce qui suggère que les performances individuelles d'agents sont corrélées à leurs performances en collectivité.

4.5.3 Grille

Le problème pour la grille est celui du calcul de gradient dans une grille de 8 par 8. La simulation dure 100 pas de temps et ne contient que deux événements : au pas de temps 1, la source d'une case passe à 8 et au pas de temps 51, cette même source revient à 0, les autres sources restant à 0. Ce scénario évalue donc la capacité d'une solution à diffuser un gradient puis à revenir à l'état initial quand la source a disparu.

Le critère d'arrêt est le suivant : un agent heuristique ne peut pas dépasser un total 1000 appels au testeur, il ne peut pas non plus dépasser 500 appels sans avoir progressé.

Heuristiques isolées

Nous présentons ici les résultats concernant les heuristiques évaluées séparément.

classe	moyenne (écart type)
Recuit	15.050 (43.861)
Genetic	36.750 (154.145)
Direct	451.190 (101.608)
Exhaustif	466.910 (320.892)
Tabou	483.760 (188.693)

Les heuristiques montrant les meilleures performances ici sont précisément celles qui semblaient les plus faibles pour l'exemple du backgammon. Pourtant les deux groupes d'heuristiques semblent se maintenir.

Populations homogènes

Nous donnons les résultats des populations homogènes.

Recherches en parallèle Nous présentons les résultats des recherches effectuées sans communication entre agents durant leurs recherches.

classe	1	4	5
Recuit	15.050 (43.861)	55.500 (89.971)	79.980 (118.681)
Genetic	36.750 (154.145)	63.670 (267.966)	61.620 (134.084)
Direct	451.190 (101.608)	542.520 (136.053)	568.600 (177.492)
Exhaustif	466.910 (320.892)	514.030 (136.194)	533.880 (166.311)
Tabou	483.760 (188.693)	568.000 (260.496)	548.210 (125.485)

L'ensemble de ces résultats suggère que la mise en parallèle de plusieurs occurrences d'une même heuristique apporte un gain.

Nous retrouvons certains résultats contre-intuitifs qui sont encore une fois les conséquences des sources de hasard embarquées dans notre système.

Recherches en concurrence La première colonne est ici identique au cas parallèle puisqu'il n'y a pas de différence entre les deux cas pour une population avec un seul agent.

classe	1	4	5
Recuit	15.050 (43.861)	128.440 (161.475)	113.980 (154.566)
Genetic	36.750 (154.145)	186.190 (608.219)	108.690 (425.288)
Direct	451.190 (101.608)	456.430 (112.236)	451.470 (305.689)
Exhaustif	466.910 (320.892)	456.580 (252.713)	427.150 (75.996)
Tabou	483.760 (188.693)	462.600 (133.535)	446.870 (101.394)

Pour Recuit et Genetic, il semblerait que l'interaction soit constructive puisqu'on observe un gain du concurrent sur le parallèle ; à l'inverse l'interaction semble négative puisqu'on observe une perte de la concurrence face au parallèle.

Ici on peut raffiner la question levée plus tôt, en effet on constate que le nombre optimal d'agents n'est pas le même pour une heuristique que ce que nous avions vu pour le testeur de backgammon.

Populations hétérogènes

Nous étudions les populations hétérogènes pour le testeur de la grille.

Recherches en parallèle Nous donnons ici les résultats pour des populations hétérogènes en parallèle.

classe	Seul	Sans	Tous
Recuit	15.050 (43.860)	515.740 (169.087)	563.180 (265.420)
Genetic	36.750 (154.145)	538.540 (244.561)	563.180 (265.420)
Direct	451.190 (101.608)	488.240 (119.382)	563.180 (265.420)
Exhaustif	466.910 (320.891)	494.240 (116.362)	563.180 (265.420)
Tabou	483.760 (188.693)	560.090 (388.903)	563.180 (265.420)

Ces résultats servent de référence pour la section suivante.

Recherches en concurrence Ici nous envisageons les populations dans le cas concurrent.

classe	Seul	Sans	Tous
Recuit	15.050 (43.860)	455.530 (219.349)	445.410 (149.097)
Genetic	36.750 (154.145)	422.330 (80.292)	445.410 (149.097)
Direct	451.190 (101.608)	453.140 (167.281)	445.410 (149.097)
Exhaustif	466.910 (320.891)	443.570 (142.255)	445.410 (149.097)
Tabou	483.760 (188.693)	448.380 (246.415)	445.410 (149.097)

On observe un gain général du cas parallèle sur le cas concurrent. Ceci indique que l'apport d'information supplémentaire en cours de recherche peut dégrader la qualité de recherche. Il est donc possible d'observer des synergies, des interactions positives mais également l'inverse.

Bilan des synergies d'heuristiques

Le premier point que nous souhaitons pointer ici est la possibilité d'étudier une heuristique sous un angle social. Il s'agit du fondement de la synergie de recherche heuristique mais elle permet également d'autres applications intéressantes. Par exemple, nous avons vu que certaines heuristiques souffrent de l'interaction : on peut interpréter ceci par une prédominance de l'exploitation sur l'exploration donnant ainsi un point de vue nouveau sur l'étude de ce compromis classique en recherche heuristique.

Un point intéressant est le nombre optimal d'heuristiques dans une population de recherche homogène concurrente : nous avons vu que ce nombre change selon les heuristiques et les problèmes. L'exploration de ce nombre

peut également fournir un nouvel élément de caractérisation d'une heuristique dans un cadre collectif.

Afin de continuer nos expériences, nous devons choisir des populations qui seront utilisées par la suite. Pour chaque problème, nous retenons la meilleure population ainsi que la population constituée d'un agent de chaque classe parallèle ou concurrente. Ainsi nous pourrons comparer pour chaque problème l'impact d'une possible synergie de problèmes sur trois populations de référence.

4.6 Synergie de problèmes

Cette section contient la description de résultats sur une recherche successive sur un problème voisin puis sur le problème principal. Nous cherchons à établir si l'on peut observer des synergies entre ces deux problèmes. L'objectif est de montrer la possibilité qu'une bonne solution à un problème voisin soit un bon point de départ pour le problème principal.

Nous explorons plusieurs possibilités :

- le problème principal est exploré deux fois de suite
- un testeur constant est proposé avant le testeur principal qui provoque une recherche aveugle
- une recherche simple sur le testeur principal sans problème initial
- une recherche sur le problème voisin avant la recherche principale

4.6.1 Backgammon

Nous décrivons tout d'abord les problèmes distincts que nous avons considérés pour le cas du backgammon.

Rappel du testeur principal Le problème principal consiste à gagner le maximum de parties parmi 7 contre un adversaire donné. Dans ce cadre, nous considérons que la qualité d'un joueur est $\frac{g-p}{n+1}$ soit le nombre de gains g moins le nombre de pertes p rapporté au nombre de parties nulles n. C'est cette valeur que le testeur principal attribue au réseau de neurones.

Testeur de distance Le testeur de distance se fonde sur un indicateur classique du jeu de backgammon qui permet d'évaluer les situations respectives des joueurs. Pour une situation donnée, on somme sur l'ensemble des pions la distance qui le sépare de l'arrivée : cela fournit la distance δ_i restant à parcourir pour les pions du joueur i. Il est commun d'accepter que le

joueur ayant la plus petite distance en cours de partie a l'avantage puisque le gain correspond à une distance nulle. Ce testeur joue les rencontres mais au lieu de compter les victoires, il enregistre les distances des joueurs en fin de partie. L'idée étant que plus la distance est faible, meilleure est la situation, la valeur du réseau de neurones étant la différence entre la distance du joueur de référence et celle du joueur utilisant ce réseau sur l'ensemble des parties jouées.

Testeur mixte distance Le testeur mixte correspond à une approche courante en intelligence artificielle quand la fonction d'évaluation correspondant au problème principal n'a que quelques valeurs qui guident peu la recherche. On imagine alors qu'une autre information pourrait aider cette recherche pour des portions de l'espace où le testeur principal donne toujours la même réponse. Dans notre cas il s'agit du testeur de distance. Afin de combiner ces deux informations, on souhaiterait que l'ordre principal soit conservé tout en bénéficiant de l'information supplémentaire : la solution naturelle est alors l'ordre lexicographique.

Résultats Nous donnons les résultats pour les trois populations que nous avons conservées. Chaque population a effectué une recherche sur chacune des combinaisons de problèmes que nous avons décrites :

- Aucun : une unique recherche sur le problème principal est effectuée ;
- Constant : une recherche sur un testeur constant est effectuée avant la recherche sur le testeur principal, ce testeur constant qui ne distingue aucun point produit une recherche aléatoire ;
- Distance : une recherche où la distance est prise en compte plutôt que le gain précède la recherche où l'on considère le gain ;
- Même : deux recherches successives sur le testeur principal ont lieu ;
- Mixte : une recherche utilisant le testeur mixte décrit ci-dessus est effectuée avant la recherche sur le testeur principal.

classe	Tous en concurrence	Tous en parallèle	5 Recuit concurrents
Aucun	2.345 (1.925)	1.650 (1.225)	3.680 (2.448)
Constant	3.227 (2.413)	2.714 (1.927)	3.893 (2.458)
Distance	4.394 (2.383)	4.459 (2.408)	4.635 (2.560)
Même	4.705 (2.408)	4.593 (2.498)	4.856 (2.551)
Mixte	4.728 (2.568)	4.372 (2.448)	4.959 (2.558)

Trois questions peuvent se poser à ce stade de nos expérimentations. L'impact du nouveau problème, l'impact de la population de recherche et leur interaction. Le premier élément que nous observons est que le problème

d'aide ne change pas la meilleure population qui reste la population de 5 agents de classe Recuit (recuit simulé) sur tous les problèmes, en général le problème d'aide ne change pas le classement entre populations.

Les deux problèmes dont on souhaitent savoir s'ils aident à la recherche principale permettent un gain par rapport à une recherche aveugle. Toutefois ces deux problèmes ne présentent pas de gain par rapport à deux recherches sur le problème principal ce qui réduit leur intérêt.

Les résultats montrent que le meilleur système est maintenant celui qui combine une recherche sur notre testeur mixte puis sur le principal. Toutefois, le gain observé sur la duplication du problème principal est faible.

4.6.2 Grille

Nous décrivons le problème alternatif spécifique à l'exemple de la grille.

Rappel du problème principal Le problème principal correspond à la recherche d'une fonction permettant à une grille d'agents de calculer collectivement un gradient. Dans le scénario du problème principal, la grille est soumise à deux phases : la première moitié sur 50 pas de temps où un signal existe et où les cellules doivent diffuser son gradient puis une seconde où le signal n'existe plus et où les cellules doivent revenir à un gradient nul.

Première moitié du scénario Dans les approches de simulations, l'approche générique permettant de définir un problème plus simple est le plus fréquemment l'utilisation d'un préfixe du scénario. Dans notre prototype, nous avons cherché si le problème correspondant à la première moitié de notre scénario principal peut aider à la recherche globale. Il s'agit donc pour notre scénario d'arriver à calculer le gradient initialement sans avoir à revenir à la situation initiale avec toutes les sources à 0.

Résultats Nous rappelons les différentes combinaisons de problèmes envisagées :

- Aucun : une unique recherche sur le problème principal est effectuée ;
- Const : une recherche sur un testeur constant est effectuée avant la recherche sur le testeur principal, ce testeur constant qui ne distingue aucun point produit une recherche aléatoire ;
- Moitié : une recherche sur la première moitié du scénario précède une recherche sur le scénario complet ;
- Même : deux recherches successives sur le testeur principal ont lieu.

Classe	Tous en Concurrence	Tous en Parallèle	5 Direct concurrents
Aucun	445.410 (149.097)	523.180 (265.420)	568.600 (177.492)
Const	893.270 (950.948)	1189.100 (792.713)	578.930 (241.261)
Moitié	1176.860 (948.405)	1719.100 (863.619)	1614.592 (880.832)
Même	1036.890 (1127.055)	1807.040 (1211.467)	618.680 (390.345)

Les résultats sont ici très différents du premier cas. Il apparaît que le problème voisin aide également la recherche sur le problème principal. Toutefois, on voit qu'une double exploration sur le même problème donne de médiocres résultats.

Cette anomalie semble justifier le choix d'explorer les synergies de recherches à travers un panel de population puisqu'on observe que la meilleure population sur une combinaison de problèmes n'est pas celle sur le problème principal.

Bilan de synergie de problèmes

Notre étude est une tentative de distinguer le réel intérêt d'un problème alternatif à travers sa comparaison avec une recherche aléatoire initiale ou encore une autre recherche sur le problème principal. L'intérêt final de ce type d'approche est la recherche de problèmes offrant un gain systématique dans la recherche sur un problème principal. Notre contribution ici est de fournir des raffinements dans la manière d'étudier cette possibilité à travers l'exclusion d'un gain aléatoire ou d'un gain comparable à une double recherche.

Nous avons également défini un moyen plus raffiné de détecter l'intérêt d'un problème d'aide dans deux cadres : le premier en construisant un testeur composite fondé sur l'idée d'ordre lexicographique et le second fondé sur la succession de deux recherches. Ainsi on peut maintenant comparer l'approche lexicographique que l'on peut retrouver chez Luke [LP02] avec une approche où les problèmes sont examinés en séquence [GM97].

4.7 Synergie Agents-Utilisateur

Cette section contient les résultats de nos expériences concernant l'imitation d'un comportement fourni par l'utilisateur avant la recherche principale. La mise en oeuvre correspond à ce que nous avons vu précédemment entre deux problèmes mais cette fois le premier problème exploré est défini à partir d'un comportement fourni par l'utilisateur.

Pour les deux testeurs, nous comparons les résultats de recherche entre une recherche simple et le résultat d'une recherche avec comme point de

départ un réseau de neurones résultant d'une recherche cherchant à imiter un comportement fourni. Pour les deux problèmes la même population de recherche est utilisée.

4.7.1 Grille

Nous rappelons ici le comportement que nous fournissons pour le calcul de gradient :

$a_{i,j}(t+1) = max(e_{i,j}(t), max(\{a_{k,l}\}_{i-1 \leq k \leq i+1, j-1 \leq l \leq j+1}) - 1)$. A partir de ce comportement, nous cherchons donc une solution qui l'imite sur le scénario du testeur principal. Ce comportement a une valeur largement supérieure à celles de toutes les solutions trouvées au cours des expériences précédentes.

classe	Tous concurrents	Tous parallèles	5 Direct parallèles
Aucun	445.410 (149.097)	523.180 (265.420)	568.600 (177.492)
Même	1036.890 (1127.055)	1807.040 (1211.467)	618.680 (390.345)
Imitation	1153.520 (945.451)	1165.700 (1038.622)	1319.670 (1025.371)

L'étude de cette synergie pose un problème important puisqu'elle dépend du comportement fourni. Pour nos deux testeurs, les comportements fournis sont meilleurs que toutes les solutions. Ici l'imitation semble être une piste intéressante mais il est impossible de dire qu'une synergie de recherche a eu lieu car les solutions trouvées sont toujours inférieures au comportement fourni. Ceci s'explique par le fait qu'aucune recherche d'imitation n'a réussi à imiter parfaitement le comportement fourni.

4.7.2 Backgammon

Nous donnons les résultats suite à l'imitation d'un comportement gagnant les 7 parties contre la fonction de référence.

classe	Tous concurrents	Tous parallèles	5 Recuit concurrents
Aucun	2.345 (1.925)	1.650 (1.225)	3.680 (2.448)
Même	4.705 (2.408)	4.593 (2.498)	4.856 (2.551)
Imitation	4.772 (2.560)	4.653 (2.402)	4.512 (2.433)

Nous observons, un gain comparable à une double recherche sur le problème principal. Toutefois, il n'y a pas domination de l'un sur l'autre. On obtient au cours de la recherche des solutions gagnant les 7 parties mais de la même manière que précédemment, l'imitation n'est jamais parfaite.

4.7.3 Bilan synergie Agents-Utilisateur

La principale leçon de nos expériences sur cette possibilité de synergie est que notre système en l'état ne présente pas de capacités d'imitation suffisante pour pouvoir réellement décider d'une synergie. En effet, il serait préférable de pouvoir comparer des résultats sur des solution qui imitent parfaitement les comportements fournis pour voir si le système pourrait ensuite les dépasser.

4.8 Conclusion

Nous avons présenté l'illustration de notre proposition à travers l'utilisation de multiples déclinaisons de notre système de recherche heuristique. Cet exposé a permis de détecter les cas où la concurrence semble favoriser la découverte d'une bonne solution mais nous avons également observé que parfois le collectif a un impact négatif sur cette performance.

Impact de l'aléatoire

Notre modèle contient deux sources principales de non-déterminisme : l'activation des agents et les appels à des générateurs pseudo-aléatoires dans les agents. Nous suggérons une extension de notre modèle pour pouvoir distinguer ces deux éléments.

Il est possible qu'un agent effectue plusieurs traitements pendant qu'un autre n'en effectue qu'un. Notre modèle permet d'envisager la possibilité de synchroniser les agents par l'introduction d'un temps discret pour le système : on autorise alors un seul traitement par agent par pas de temps[14]. Cette modification permet alors d'avoir un modèle multi-agent déterministe[15] et de circonscrire l'aléatoire à l'intérieur des agents. Toutefois, cette solution crée un déséquilibre entre les agents effectuant un traitement lourd sur un pas de recherche (Genetic) et ceux ayant un pas de recherche plus léger (Recuit).

Etude sociale d'heuristiques

Les expériences que nous avons menées nous permettent d'envisager une étude des heuristiques de manière sociale. L'ensemble des résultats

[14]Seuls les agents ayant des messages à traiter sont concernés.

[15]Un autre élément nécessaire est une fonction interclassant les messages émis par les différents agents sur un pas de temps.

que nous avons rencontrés montre qu'il est important de considérer trois éléments : l'heuristique, le problème et l'interaction à travers la distinction entre concurrence et parallélisme. Nous avons donné des résultats en terme de meilleure solution mais la dimension collective de la recherche peut également être envisagée à travers des mesures collectives comme le nombre de messages ou encore la qualité des solutions contenues dans les messages comparée à celles contenues dans les mémoires.

Nombre optimal

Nos travaux sur les synergies d'heuristiques ont montré qu'un nombre optimal d'agents dans une population d'agents existe sans doute pour une grande variété de combinaisons heuristique/problème. L'étude plus approfondie de ce nombre présente un intérêt majeur envisagée dans le cadre de la multiplication des architectures multi-machines, muti-processeurs et multi-coeurs. Elle implique en particulier que l'augmentation du nombre d'agents n'est pas toujours bénéfique.

Combinaison de problèmes

Nous avons cherché à mettre en place les outils pour une étude rigoureuse de l'intérêt de combiner plusieurs problèmes dans un cadre de recherche heuristique. Nous avons envisagé les moyens de dissocier le simple intérêt d'une recherche aléatoire mais également une double recherche sur le problème principal, deux possibilités qui relativisent l'intérêt d'un problème voisin.

Co-construction

La co-construction de solution par un système intelligent est un objectif séduisant. Nous avons cherché à mettre en place un cadre permettant d'évaluer l'intérêt de l'imitation de l'utilisateur par le système. Pourtant, les résultats ne sont pas exploitables à cause de la faible capacité d'imitation du système. Par ailleurs, l'influence ne va pour l'instant que de l'utilisateur vers les agents.

Chapitre 5

Emergences a posteriori

Jusqu'ici nous avons cherché à produire une approche qui permette d'envisager une supériorité du tout sur la somme de ses parties. Nous revenons dans ce chapitre sur les éléments du modèle qui permettent cette distinction, qui nous permettent de définir l'émergence dans un système multi-agent.

5.1 Retour formel sur notre modèle

Un SMA est un ensemble d'agents, chacun responsable de sa mémoire, percevant et envoyant des messages[1]. Ces entités s'exécutent de manière simultanée ce qui implique une part de non déterminisme. Le vie d'un agent est une succession de traitements qui consomment un message et sont l'unique occasion d'un changement de la mémoire d'un agent et de l'envoi de messages. Dans le reste de ce chapitre, l'indice est une lettre grecque et correspond à un agent alors que l'exposant est une lettre latine et correspond au temps.

5.1.1 Relation d'englobant

Nous avons défini l'organisation d'un système à travers une relation qui décrit le fait qu'un agent est le père d'un autre. Cette relation que nous notons O(pour organisation) définit un arbre d'agents doté d'une racine. Cet arbre induit un ordre : pour deux agents α et β $\alpha > \beta$ si et seulement si le chemin de l'arbre de la racine à β passe par α. Autrement dit, α est un ailleul de β.

[1]Dans Ferber [Fer95], on qualifie de tels agents de "purement communiquants".

Notre relation sert à limiter les envois de messages. Un agent α ne peut envoyer un message à un agent β que si $\alpha O \beta$ ou $\beta O \alpha$, ce qui inclut son père ainsi que ses fils. Il peut aussi s'envoyer un message. L'ordre induit nous permet de dire qu'un message est ascendant si son expéditeur est supérieur à son destinataire, descendant sinon.

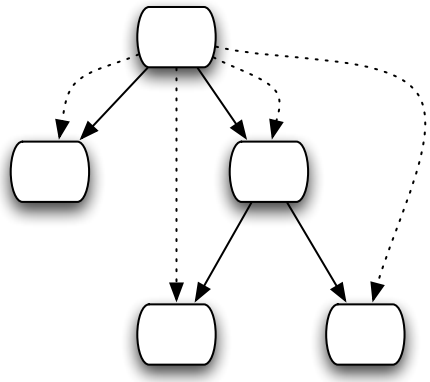

FIG. 5.1 – Relation d'ordre partiel entre les agents en pointillé. En trait plein la relation "père de".

Finalement, deux agents α et β sont au même niveau s'il existe un agent γ auquel ils sont tous deux supérieurs et que les longueurs des chemins de γ à α et de γ à β sont égales.

5.1.2 Interactions

Les agents communiquent à travers des envois de messages. Les envois de messages ne sont pas bloquants et la réception se fait à travers une file de messages par agent.

L'ensemble de l'exécution du système peut être envisagée sous l'angle de l'interaction à travers les messages échangés dans le système. La réception et le traitement d'un message provoquent l'envoi de nouveaux messages. Ainsi ce modèle où un message peut en causer un autre nous permet de mettre en place une relation reliant les messages entre eux.

Plus concrètement, une exécution d'un SMA donne lieu à l'échange d'un ensemble de triplets constitués chacun d'un expéditeur, d'un message et d'un

destinataire $(de, message, vers)$[2]. La relation $\rightarrow$ désigne alors le fait qu'un message en a provoqué un autre à travers le traitement qu'il a provoqué chez l'agent destinataire, on note ainsi $(\alpha, x, \beta) \rightarrow (\beta, y, \gamma)$ le fait que la réception du message x envoyé par α à β a provoqué un traitement produisant le message y à destination de γ. En exemple, nous donnons une recherche simple d'un seul agent heuristique (cf. figure 5.2).

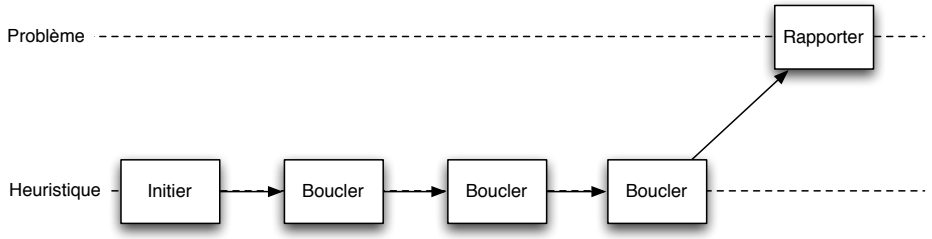

FIG. 5.2 – Une unique recherche, les messages sur la même ligne sont traités par le même agent.

Cette relation embarque l'information nécessaire à la compréhension de l'exécution du système et permet également de se donner une représentation visuelle à travers le graphe correspondant. On note qu'il s'agit alors d'un graphe orienté, acyclique car un message ne peut pas à la fois provoquer et être provoqué par un autre. Cette relation peut également être comparée à la relation "happened before" de Lamport [Lam78].

5.1.3 Dynamique Agent

L'ensemble des agents est noté A. Le système modifie l'ensemble de mémoires noté M, la mémoire de l'agent α est notée M_α. Une classe d'agent correspond à la définition d'un type de mémoire et des fonctions de traitement permettant à l'agent de traiter les messages qu'il reçoit.

Une fois recentré sur l'agent, notre modèle implique une séquence des mémoires successives d'un agent, dont les transitions sont étiquetées par les messages reçus. Cet historique de l'agent permet de rejouer l'exécution de

[2]Deux messages sont égaux s'ils sont le même message et non s'ils ont le même contenu. Ainsi deux messages distincts, s'ils ont même contenu, même expéditeur et même destinataire forment deux éléments de l'ensemble des triplets.

l'agent isolé. Graphiquement, on peut se représenter l'exécution du système en combinant graphe de messages et séquence des mémoires (cf. figure 5.3).

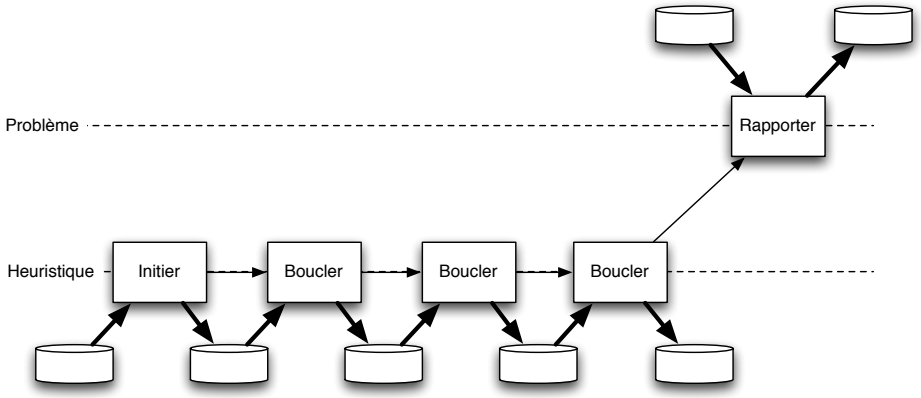

FIG. 5.3 – Une unique recherche où les transitions entre mémoires sont étiquetées par les messages.

5.1.4 Dynamique gros grain

Nous introduisons ici la notion de temps nécessaire à la description de la dynamique du système. L'état du système est complètement défini par les mémoires des agents M^t et leurs files de messages F^t. La dynamique du système correspond aux traitements des messages contenus dans les files des agents. Le système passe d'un état $S^t = F^t \times M^t$ à l'état S^{t+1} par l'application d'une fonction de traitement sur le premier message d'une file non vide. Cette fonction associe à la mémoire antérieure de l'agent M^t_α et au message entrant la mémoire postérieure M^{t+1}_α et une liste de messages à envoyer placés dans les files des agents concernés formant F^{t+1}. Dans le cas où plusieurs files d'agent sont non vides, l'exécution du système suit l'un des chemins possibles[3] et devient donc non déterministe.

5.1.5 Dynamique grain fin

Nous n'incluons pas dans le modèle ce qui se passe au cours du calcul d'une fonction de traitement. Dans le cadre de la thèse, nous avons utilisé

[3]Le futur du système dépend alors des temps d'exécution des traitements de ces agents.

plusieurs formalismes pour représenter ces fonctions : un langage pseudo-algorithmique dans le manuscrit et le langage Java dans l'implémentation. Ainsi notre modèle est paramétrable par un modèle de calcul.

5.2 Emergence et synergies

Le principal objectif de notre modèle et de notre système est de définir précisément la notion de synergie entre agents. C'est dans cet optique que nous avons défini notre système de recherche heuristique multi-agent.

5.2.1 Impact du nombre

Une première piste dans l'étude de l'intérêt collectif d'un système multi-agent est naturellement l'étude de l'impact du nombre d'agents. Cette étude se mène alors sur des populations homogènes qui ne se différencient que par le nombre d'agents qu'elles contiennent.

Dans nos expériences, nous avons évalué l'impact du nombre de différentes manières : dans le cadre de populations de recherche homogènes, nous avons vu que si l'ajout d'agents ne peut pas dégrader une recherche parallèle[4], le fait de rajouter un agent peut être négatif dans le cadre d'une recherche concurrente. Dans le cadre d'une synergie de problèmes, nous avons étudié l'impact de l'ajout d'un agent, cette synergie joue donc également sur le nombre d'agents.

Parmi les travaux existants, la proposition de Kubik implique que l'émergence est reliée au nombre d'entités du système. Pourtant, nous avons vu que si toutes ces entités sont de la même classe, l'émergence telle que Kubik la définit n'a pas lieu. L'intérêt de notre proposition est donc de fournir une définition de l'émergence entre heuristiques homogènes que l'on peut étudier en fonction du nombre d'agents.

5.2.2 Exécution parallèle ou concurrente

Une des caractéristiques intéressantes de notre système, en particulier dans le cas d'une synergie d'heuristiques, est la possibilité d'envisager les exécutions parallèles et concurrentes. Pour deux systèmes très proches, nous obtenons le moyen d'attribuer le gain ou la perte de performance à l'interaction supplémentaire.

De manière informelle, le cas d'une recherche concurrente est celle où les messages d'une heuristique peuvent influencer la recherche d'une autre.

[4]Si ce n'est en termes de temps d'exécution.

Cette influence n'est pas automatique puisque l'agent peut choisir de ne pas en tenir compte mais cette propriété est une condition nécessaire à l'échange d'informations entre les agents. A l'inverse, on a le cas où aucune information émise par une heuristique ne peut parvenir à une autre heuristique pendant sa recherche.

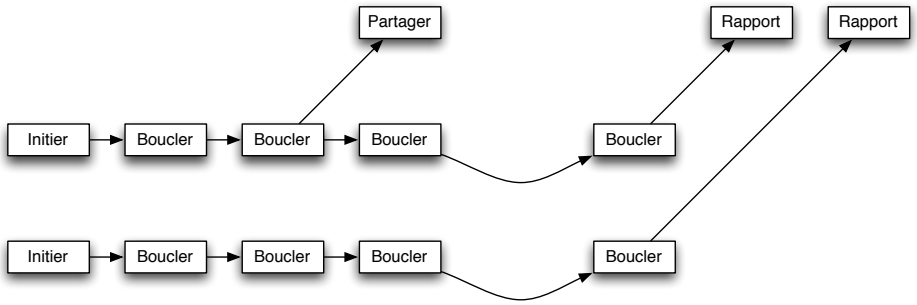

Fig. 5.4 – Une recherche avec deux heuristiques en parallèle

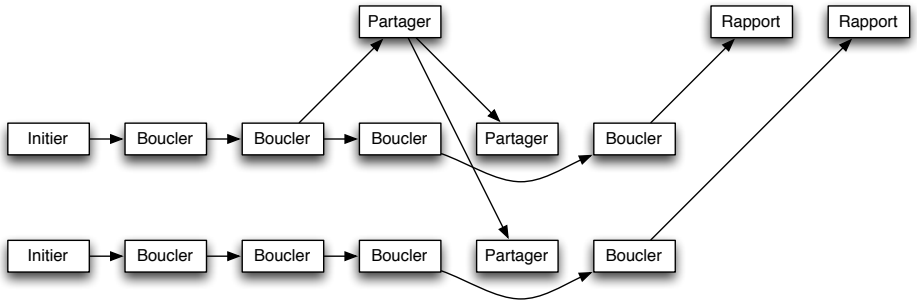

Fig. 5.5 – Une recherche avec deux heuristiques en concurrence

Le cas concurrent est celui où il existe un chemin dans le graphe d'interaction reliant les agents de la population. Le cas parallèle nous indique qu'il n'est pas possible que les agents se soient influencés durant leur exécution.

Notre modèle permet donc de déterminer si différents agents ont pu s'influencer à partir de l'analyse du graphe d'interaction résultant de l'exécution.

Nous pensons qu'en dehors de notre système où nous avons délibérément cherché cette propriété, ce type d'outils d'analyse peut servir dans de nombreuses applications multi-agent afin d'identifier les agents fortement liés par l'interaction ou encore de délimiter l'ensemble des agents potentiellement influencés par un événement.

Cette vision de deux modes d'exécution est proche de celle de la théorie des AMAS qui distinguent les SMA coopératifs de ceux présentant des situations non coopératives. Les deux approches consistent à étudier une propriété de l'exécution du système et plus particulièrement des interactions. La principale différence est qu'un agent d'un AMAS[5] cherche systématiquement à éviter les situations non coopératives ce qui implique une seule exécution du système. Notre proposition permet d'obtenir deux exécutions dont on peut comparer les qualités et propriétés alors que les AMAS présente l'avantage de s'adapter en cours d'exécution.

5.2.3 Importance de l'objectif

Une des phrases les plus présentes dans la littérature concernant l'émergence est "le tout est supérieur à la somme des parties". Cette phrase contient deux idées importantes : le tout et la somme des parties sont deux choses différentes mais surtout le tout est supérieur. Pour Corning, il s'agit d'obtenir des effets différents selon que le système est composé ou non. Dans le travail de Kubik, il s'agit d'une inclusion stricte entre deux ensembles. Dans notre système, nous avons cherché à aborder un domaine dans lequel l'objectif du système est facile à identifier puisqu'il est modélisé à travers une fonction. Nous avons donc cherché à nous placer dans un cadre où nous pouvons affirmer que le tout est bien supérieur à la somme de ses parties et pas seulement différent.

Dans la méthodologie ADELFE, la satisfaction de l'objectif est assimilée à l'obtention d'un comportement collectif. Nous avons cherché à distinguer ces deux idées pour chercher une corrélation expérimentale. La combinaison de ces deux approches peut se révéler constructive en choisissant de maintenir la coopération dans le système quand celle-ci mène clairement à un meilleur fonctionnement ou à une meilleure solution tout en permettant des modes parallèles dans les cas où l'intérêt de la coopération est nulle voire négative.

[5] Adaptive Multi-Agent System.

5.3 Influence entre les niveaux

Une des caractéristiques principales de l'émergence consiste à placer les entités/propriétés à différents niveaux. Cette caractéristique très répandue impose d'avoir une notion claire de ce que sont les niveaux dans le système afin de pouvoir déterminer ce qu'est l'émergent.

5.3.1 Agents et Agents

L'arbre des agents nous permet naturellement de définir des niveaux auxquels se situent les agents. Cette possibilité en plus de l'analyse du graphe d'interactions permet au cours d'une exécution du système de déterminer s'il y a eu influence réciproque entre des agents de différents niveaux comme dans le cas d'une recherche concurrente.

Nous remarquons également que le graphe d'interactions nous permet de déterminer si des agents ont pu s'influencer réciproquement mais ne permet pas de savoir si cette influence a eu lieu. De manière générale, l'autonomie de l'agent nous interdit d'aller plus loin car il est impossible de savoir si l'agent a extrait de l'information des messages reçus sans inspecter sa séquence de mémoires. A l'inverse, le graphe d'interaction peut permettre d'exclure la possibilité d'une influence réciproque entre deux agents.

Cette possibilité rejoint la proposition de Sawyer où des entités de différents niveaux s'influencent réciproquement à travers des interactions ascendantes et descendantes. On retrouve l'idée d'une entité de haut niveau dont les propriétés sont déterminés par l'activité de ses composants et exerçant une influence sur ces composants.

5.3.2 Vrai composite

Nous avons cherché à construire un modèle multi-agent dans lequel on peut considérer qu'un agent composite (SMA) est un agent au même titre qu'un agent atomique. Cette conception nous permet naturellement de placer les agents à différents niveaux selon leur place dans l'arbre de l'organisation du système. Un agent problème est un agent composite.

Une des questions sous-jacentes de l'émergence est la définition de vrai composite. Nous rappelons la proposition de Searle : on a émergence 1 si une phénomène n'est explicable qu'à travers la prise en compte des interactions entre les parties du système et pas uniquement de la composition du système.

Ce rappel nous permet d'envisager un agent problème dont la recherche se fait de manière concurrente comme un vrai composite : en effet, la com-

paraison avec le système correspondant en parallèle indique que la différence ne vient que d'une différence d'interaction. Le gain ne peut s'expliquer par la composition du système puisque système concurrent et parallèle sont composés des mêmes agents. Ainsi le gain ou la perte de performance du système concurrent est une propriété émergente 1.

5.3.3 Organisation et Agents

Nous finissons cette discussion sur les niveaux d'un système par un élargissement. Jusqu'ici nous n'avons traité que d'agents mais il est intéressant d'envisager les influences possibles entre l'organisation du système et les agents.

En effet, l'organisation est une entité du système, qui existe et qui contraint l'activité des agents. Dans notre système, l'organisation ne varie pas mais dans le cas d'une organisation dynamique, il est possible d'imaginer une influence réciproque entre l'organisation et les agents.

5.4 Ce qui émerge

Finalement, nous revenons sur une question évidente : qu'est-ce qui émerge ? Cette question simple n'est pas toujours traitée dans les définitions d'émergence : parfois le système est qualifié d'émergent sans que l'on puisse dire ce qui "émerge".

Le résultat du système Dans le cadre de notre système, nous avons toujours distingué plusieurs versions d'un système pour ensuite comparer leurs résultats : les solutions produites. L'approche la plus naturelle est de considérer que ce qui émerge sont les solutions produites par une exécution du système.

On peut alors dire qu'une bonne solution a émergé de l'interaction entre plusieurs agents au cours de l'exécution du système.

L'émergence a lieu au cours de l'exécution d'un système. Le meilleur point de la recherche émerge de l'influence réciproque et constructive entre agents de différents niveaux. Il s'agit de l'approche suivie par Kubik qui qualifie d'émergents les produits du système.

Le système lui-même Au cours des différentes variations de notre système de recherche, nous avons vu apparaître des interactions positives aidant à la résolution du problème soumis au système. Ceci suggère que le

caractère émergent du résultat réside dans la comparaison entre le système synergique et le système de référence.

On peut alors considérer que ce qui émerge est le système de recherche soit la métaheuristique composée par un système multi-agent dans lequel l'interaction provoque un gain de performance à travers une exécution concurrente. Cette possibilité rejoint l'idée de vrai composite dont les propriétés ne peuvent être attribuées qu'aux interactions dans le système.

L'émergence a lieu entre différentes versions d'un même système. Par extension, les solutions déterminées par un tel système peuvent ensuite être qualifiées d'émergentes. Ce point de vue est partagé par la théorie des AMAS dans laquelle l'émergence a lieu si les agents maintiennent une activité coopérative : le résultat peut alors être qualifié d'émergent.

5.5 Conclusion

Dans ce chapitre, nous sommes revenus sur les principales caractéristiques de l'émergence que nous avons favorisées au cours de cette thèse : les synergies dans un modèle multi-agent. Nous avons justifié l'appellation émergente mais nous avons surtout souhaité mettre en avant les outils formels que les caractéristiques de notre modèle permettent.

Chapitre 6

Retour sur les émergences

Au cours des précédents chapitres, nous avons concentré nos efforts autour des idées de synergies et de niveaux dans un modèle d'agents hiérarchique. Dans ce chapitre, nous explorons les possibilités que notre modèle peut fournir dans le cadre d'autres approches de l'émergence. Il ne s'agit que de pistes qui ne constituent pas notre travail principal mais nous pensons qu'elles peuvent suggérer comment un modèle multi-agent peut servir de socle à la définition de notions associées à l'émergence et donc à d'autres visions de l'émergence.

6.1 Inscription et Interprétation

L'émergence à travers une possibilité d'inscription puis d'interprétation est une possibilité que l'on retrouve dans le travail collectif [tc97] ou encore dans la proposition de Muller [Mul03]. L'inscription se fait habituellement dans un environnement partagé par les agents : nous considérons que le père d'un agent peut être vu comme son environnement : par exemple, l'agent problème qui englobe les agents heuristiques peut être vu comme leur environnement dans lequel ils perçoivent et agissent.

Inscription et interprétation sont les conditions de l'émergence [tc97, page 14] :

> "Cette observation ne peut se faire qu'à travers une inscription du phénomène d'une part et une interprétation de cette inscription par l'observateur ou par le système d'autre part dans un vocabulaire ou une théorie D' distincte de D"

La condition d'inscription se transcrit en une condition de changement de mémoire pour un agent père lors de la réception d'actions de ses fils. Dans

notre système la construction d'une trajectoire par l'agent problème à partir des solutions envoyées par les agents heuristiques constitue une inscription de leurs pas de recherche.

La condition de l'interprétation est plus difficile à définir. Elle est décrite ainsi :

> "les agents ne peuvent percevoir et agir que localement dans cet environnement. Autrement dit, chaque agent interprète l'environnement suivant ses moyens limités (d'après les distinctions qu'il peut faire)."

Ce passage suggère que l'interprétation constitue une perception personnelle de l'inscription. Dans notre système, nous avons vu que les agents heuristiques traitent leur perception de la trajectoire globale selon leur propre mémoire, décidant de conserver ou d'ignorer la solution mise à leur disposition. Les agents heuristiques effectuent donc une interprétation au sens d'un traitement personnalisé du message reçu.

L'autre sens d'interprétation est le passage d'une théorie / vocabulaire / langage à un autre. On peut facilement envisager que l'observateur n'utilise pas le langage du système pour en faire une interprétation mais la possibilité que le système manipule deux langages pose problème. Nous décrivons deux possibilités sur cette question des langages.

Langages des agents Une première idée, très informatique est d'envisager la possibilité de deux langages de programmation, de description des comportements des agents. Nous avons vu que notre modèle, s'il suppose que les agent puissent échanger des valeurs à travers les messages, ne fait pas d'hypothèses sur le modèle de calcul, sur le langage de description des agents. Ainsi rien n'empêche qu'un phénomène inscrit dans le père de l'agent dans un langage soit interprété par le fils dans un autre langage.

Langages manipulés par les agents La seconde idée consiste à envisager la possibilité que les différents agents manipulent des vocabulaires différents. Dans notre système, nous avons vu que des agents peuvent manipuler des comportements ou des réseaux de neurones[1]. Dans ce sens, il est possible d'envisager notre problème d'imitation comme l'interprétation d'un comportement en terme de réseau de neurones. Cette distinction semble plus plausible que la précédente mais pose la question plus difficile de la distinction entre les deux langages manipulés dans un seul agent. En effet, si la

[1] On note au passage que ce sont précisément les exemples de langages différents utilisés dans [RS01].

distinction entre comportement et réseau de neurones parait claire, cela ne constitue pas un critère général de délimitation.

6.2 Emergence et Auto-Organisation

Un intérêt d'un modèle multi-agent dans le cadre de l'émergence est de clarifier le lien entre émergence et auto-organisation. Nous avons modélisé l'organisation comme une structure régissant les envois de messages entre agents. Dans la discussion autour de l'auto-organisation, nous avons mis en avant deux conceptions dominantes de cette notion : la première définit l'organisation comme la structure unissant les agents et l'autre comme le caractère ordonné d'une série d'observations.

6.2.1 Organisation d'un agent

L'organisation dans une série de données est la piste de travail explorée par Crutchfield, Shalizi ou encore Young. Elle consiste à étudier des mesures fondées sur l'entropie sur une séquence de symboles appartenant à un alphabet fini. Il s'agit ensuite d'utiliser ces mesures dans la reconstruction d'une machine correspondant à la séquence de symboles. L'application de ce type d'approche à l'étude de systèmes à base d'agents nous incite à chercher à reconstruire un agent à partir de sa simple observation.

Cette voie de recherche consistant à chercher à extraire la structure d'un agent à partir de son observation est très séduisante car elle permettrait de successivement remplacer des comportements observés par des machines reconstruites à partir de ceux-ci. Toutefois, les travaux existants ne concernent que des exemples simples [Cru94, Sha01]. Il est rare qu'un agent exhibe deux mémoires égales au cours d'une exécution. Dans ce cadre, la séquence de mémoires apparaît donc comme complètement aléatoire ce qui interdit son étude. Ce dernier point nous sert à indiquer que ce type d'approches est sans doute plus adapté à des agents dont l'espace des mémoires[2] est petit par rapport à la longueur de l'exécution ce qui semble désigner des agents réactifs[3].

[2] L'espace des états internes de l'agent.

[3] On note qu'un exemple d'application de la thèse de Shalizi concerne des automates cellulaires que l'on peut considérer comme des agents réactifs.

6.2.2 Organisation d'une structure d'agents

Dans notre modèle multi-agent, nous avons travaillé sur des agents structurés par un arbre et cet arbre ne changeait pas au cours de l'exécution du système. L'auto-organisation nécessite une évolution de l'organisation du système au cours du temps, il est donc nécessaire de permettre à la structure d'évoluer avec le temps alors que nous avons principalement joué sur le rôle de l'interaction.

Dans notre travail sur la recherche heuristique nous n'avons pas traité de la modification de cet arbre mais cela reste possible[4]. On peut alors envisager l'auto-organisation comme l'accroissement d'une mesure observable de cet arbre comme par exemple, une entropie comme celle des systèmes multi-agent minimaux [Aek99].

Il faut déterminer plusieurs points concernant l'organisation : i) sa nature, est-ce un graphe, orienté, un arbre, un arbre à racine, un treillis, un ensemble de faits, de règles ii) son évolution, qui peut la modifier, comment iii) comment l'organisation augmente ou diminue. Dans notre modèle, une organisation est un arbre d'agents, les modifications possibles sont la naissance ou la mort d'un agent et les mesures possibles sont celles applicables à un arbre avec la possibilité supplémentaire de donner une valeur aux arêtes fonctions des messages y circulant[5].

Le caractère autonome d'un accroissement du caractère organisé dépend de l'ouverture à l'extérieur du système. Pourtant, il est important de noter qu'en incluant les entités en interaction on arrive toujours à un système fermé. L'utilisation conjointe de la séquence d'arbres et du graphe d'interaction peut ici permettre de délimiter des périodes de l'exécution pendant lesquelles le système est isolé de l'extérieur tout en s'organisant. On peut alors envisager la question sous la forme de l'auto-organisation d'un sous-système à travers l'évolution comparée de son organisation et de celle du reste du système clos l'englobant[6] tout en suivant finement si ce sous-système reçoit des messages externes.

Finalement, il faut noter que de nombreux travaux sur l'auto-organisation ne contiennent pas de modélisation explicite de l'organisation ou d'une mesure augmentant au fil de l'exécution. L'auto-organisation fait alors référence à une structure sociale qui fournit le support de l'exécution

[4]Nous travaillons à intégrer la naissance et la mort d'agents à la dynamique du système qui permettrait d'avoir une dynamique d'organisation.

[5]Nous rappelons que l'organisation décrit les canaux possibles pour les messages.

[6]La même idée sous-tend l'explication du démon de Maxwell : l'entropie du gaz diminue alors que l'entropie du système gaz et du démon augmente.

du système ce qui correspond à la plupart des modèles d'insectes sociaux comme les fourmis, les termites ou encore certaines espèces d'araignées. De nombreux travaux traitent de l'utilisation de ce principe d'auto-organisation pour la résolution collective de problèmes [Che04] dont les applications peuvent servir à résoudre des problèmes allant du voyageur de commerce [DS04] au partage de ressources critiques [AHP05]. Le caractère auto-organisé correspond alors à la capacité de l'organisation à résoudre le problème.

6.3 Surprise et Prédiction

Dans le cadre de notre modèle multi-agent, l'exécution d'un système peut donner lieu à trois dynamiques : l'agent produit ses mémoires successives, les messages se déroulent et l'organisation produit une séquence d'arbres. Nous cherchons ici à montrer en quoi une décomposition multi-agent peut aider à préciser les problématiques de prédiction et de surprise.

6.3.1 Déterminisme

Le premier point quand on parle de prédiction concerne le déterminisme de l'exécution du système. Dans notre modèle, le même système peut donner lieu à des exécutions différentes selon l'affectation du temps de calcul aux agents et l'ordre de livraison des messages. Ce point nous permet de mettre en avant le fait que de nombreux modèles comme le modèle d'acteurs, le modèle sous-jacent du langage Erlang ou encore notre modèle multi-agent ne garantissent pas qu'un système décrive une unique exécution.

Cette famille de modèle regroupe donc à la fois des systèmes déterministes et d'autres qui ne le sont pas, ceci dépendant des comportements des agents. De tels modèles permettent de distinguer deux classes parmi les systèmes qu'ils recouvrent : ceux ne donnant lieu qu'à une exécution possible et ceux pouvant en engendrer plusieurs. Cet élément nous permet de préciser la question de la prévisibilité d'un système multi-agent en contraste avec les travaux de Bedau [Bed97] ou Darley [Dar94] qui, traitant d'automates cellulaires, n'abordaient pas cette caractéristique.

6.3.2 Information partielle

Notre modèle, à travers la décomposition en Organisation, Interaction et Agent permet d'envisager cette question sous un angle particulier : l'idée

est de considérer le pouvoir prédictif de chacune de ces composantes sur les autres.

Cette possibilité permet notamment de raffiner une proposition de test fondé sur la surprise [RS01] : en considérant que l'observateur a accès à l'information d'interaction et aux séquences de mémoires d'agent, à quel point est-il surpris par l'organisation du système. Plus largement, la question de la surprise peut se poser pour toutes les combinaisons possibles entre i) information connue de l'observateur et ii) information à juger surprenante ou pas.

Dans le cadre de la prédiction et des possibles critères statistiques que l'on peut y appliquer, la même démarche peut s'appliquer en combinant des approches comme celle de Shalizi [Sha01] où la prédiction est évaluée sur la capacité à prédire le futur à partir du passé et la décomposition du système multi-agent selon ses organisations, interactions et mémoires d'agents successives.

6.4 Bilan

Dans ce chapitre, nous avons suggéré en quoi notre modèle d'agents permet d'éclairer certains aspects reliés à la discussion sur l'émergence. Ce travail permet d'envisager l'intérêt du travail effectué en dehors du point particulier des synergies de recherche.

Nous avons cherché ici à mettre en avant les combinaisons entre notre modèle et d'autres approches de l'émergence, avec deux objectifs : suggérer des usages possibles de notre modèle mais surtout ancrer le discours dans un modèle commun, en tenant compte de spécificités multi-agent.

Chapitre 7

Conclusion

Au cours de ce manuscrit, nous avons envisagé les liens entre émergence, systèmes multi-agent et résolution de problèmes à travers une triple restriction :

- restriction de la résolution de problèmes à la recherche heuristique
- restriction de l'émergence à la caractérisation de vrais composites par les synergies
- restriction des systèmes multi-agent à un modèle multi-agent fondé sur des traitements fonctionnels et une organisation arborescente

Nous concluons cette thèse en revenant sur ces trois points : sur les contributions ainsi que sur les perspectives que ce travail de thèse a ouvertes.

7.1 Recherche Heuristique

Notre travail dans le domaine de la recherche heuristique constitue une formulation agent de la plupart des notions du domaine dans lequel nous avons inclu plusieurs idées visant à étudier les possibilités collectives d'un système. L'idée principale est alors d'envisager l'intérêt du partage de solutions.

7.1.1 Contribution

Le premier point est le parallèle entre agent/SMA et heuristique/métaheuristique. On retrouve cette correspondance dans les travaux de Milano et Roli [MR04] qui expriment plusieurs métaheuristiques sous la forme de systèmes multi-agent. La grande différence est alors que dans leur proposition, les moyens permettant la communication entre heuristiques du-

rant leur recherche ne sont pas fournis ce qui limite l'intérêt collectif à des recherches en parallèles. Nous donnons un exemple de métaheuristique que l'on peut alors exprimer comme un système multi-agent de notre modèle : un algorithme de colonie de fourmis [DS04] est un agent père dont les fils sont les fourmis, les déplacements des fourmis modifient les niveaux de phéromones maintenus par l'agent père qui informe les fourmis de ces changements qui changent alors leurs comportements. Ce type de métaheuristique est difficilement modélisé dans le cadre de Milano et Roli où l'ensemble de la population de fourmis doit être géré par un seul agent.

Un autre intérêt de notre modèle est de permettre la définition des variantes parallèle et concurrente d'une métaheuristique : ceci ouvre la possibilité d'étudier les caractéristiques *sociales* d'algorithmes de résolution de problèmes. Cette approche nous rapproche du domaine de la recherche heuristique coopérative comme par exemple [BCK05] qui étudie les performances respectives de deux heuristiques séparées puis rassemblées en un seul système. Nous avons également défini un cadre de synergie de problèmes qui partagent des caractéristiques avec des approches d'apprentissage incrémentiel. Notre contribution consiste ici à isoler le réel intérêt de ces recherches coopératives en les confrontant à plusieurs systèmes de référence (recherche aléatoire, double recherche, populations homogènes de tailles variables).

Finalement, nous avons implémenté l'ensemble du système ce qui nous a permis d'exhiber des cas de recherche heuristique où nous pouvons attribuer un gain de performance à l'interaction. Toutefois, nous avons également pu nous rendre compte au cours de ces expérimentations que la mise au point d'un tel système nécessite du recul sur les heuristiques, leurs paramètres et les problèmes traités.

7.1.2 Perspectives

Multiples Formats, Multiples Problèmes Nous avons présenté un système effectuant une recherche dans l'espace des réseaux de neurones, mais le domaine de l'intelligence artificielle a donné naissance a de nombreux formats permettant de représenter des fonctions ainsi qu'aux algorithmes permettant leur manipulation. L'enrichissement de notre système par ces multiples formats permettrait par exemple de déterminer si certains formats sont plus adaptés à certains problèmes.

Automatisation De nombreux éléments peuvent varier dans un système : la population d'heuristiques, les paramètres de ces heuristiques, les

problèmes considérés. L'ensemble de ces possibilités est énorme, aussi il semble nécessaire d'envisager une recherche automatique du meilleur système pour un problème donné : le résultat d'un tel système pourrait alors indiquer l'intérêt d'un problème voisin ou encore la meilleure population.

Co-construction L'interaction que nous avons permise entre l'utilisateur et le système était à sens unique. Nous suggérons ici une possibilité de co-construire un comportement pour le jeu de backgammon. Il s'agirait d'envisager le processus d'imitation du comportement enregistré de l'utilisateur avec l'un des deux processus suivant :

1. Soit le joueur joue contre les solutions l'imitant le mieux, poussant ainsi le joueur à varier son jeu pour battre son propre comportement.
2. Soit le joueur confronté à une nouvelle situation reçoit les propositions des éléments l'imitant le mieux (dans le cadre du jeu contre des adversaires humains).

Ces deux possibilités seraient évaluées par leur capacité à faciliter l'apprentissage du jeu.

7.2 Synergies et Emergence

7.2.1 Contribution

Notre travail permet de mettre en perspective la question de l'existence de l'émergence. Tout d'abord l'existence dépend de la définition, ensuite l'existence peut dépendre de détails omis dans la définition. Nous avons vu que plusieurs définitions trouvent des implications différentes voire contradictoires selon les précisions qu'on y apporte. Dans notre travail sur la recherche heuristique, nous observons des synergies qui permettent d'affirmer l'émergence.

Dans le cadre de la discussion sur l'intérêt collectif, nous avons assimilé l'émergence au cas d'un tout supérieur à la somme de ses parties. Cette propriété vise alors à qualifier des vrais composites, soit des agents composites dont les propriétés ne sont pas réductibles à leur composition. Nous avons cherché à définir cette propriété à travers l'exclusion de l'influence d'une autre entité du système en définissant des systèmes qui ne diffèrent que par l'interaction. Ainsi la combinaison d'un modèle permettant la définition de composites et la caractérisation de synergies nous a permis de définir des systèmes de recherche heuristique dans lesquels une part de la qualité est attribuable au collectif.

7.2.2 Perspectives

Graphe NP et réduction Une perspective concerne la possibilité d'adresser un problème en NP à l'aide des problèmes NP auxquels il se réduit. Pour un problème NP donné et une instance donnée, on chercherait simultanément :

- des solutions à l'instance à l'aide des meilleures heuristiques pour ce problème
- des solutions à des instances des problèmes auxquels le problème de départ se réduit

L'idée consisterait alors à mettre en place un système de résolution de problèmes NP. Il est peu probable que ce système donnerait des bonnes solutions à travers une combinaison réduction/résolution pour des problèmes connus, toutefois, il semble intéressant afin d'explorer de nouveaux problèmes NP à travers le crible des heuristiques existantes pour d'autres.

7.3 Modèle multi-agent

Les agents fonctionnels tels que nous les avons définis correspondent à la gestion d'une mémoire et de messages par des fonctions au sens mathématique du terme, combinée à la gestion de messages asynchrones. Nous pensons que ce modèle constitue une approche de l'agent vu comme une extension de la fonction à travers l'ajout du passage de messages. Cette possibilité fournit une alternative à l'idée de l'agent comme une extension de l'objet[1].

7.3.1 Contribution

Nous avons exposé notre modèle ainsi qu'un certain nombre de ses caractéristiques. Plusieurs propriétés de ce modèle sont héritées de ces sources : par exemple la relation reliant les messages est similaire à la relation d'activation dans [HBS73]. Toutefois la combinaison d'une organisation hiérarchique, d'agents fonctionnels et de cette relation entre messages est une

[1] Ceci implique ma propre définition d'agent : le terme d'agent est le terme naturel pour décrire une entité en interaction avec d'autres à travers des envois de messages. Il me semble que la notion d'agent vient compléter élégamment celle de fonction en décrivant un modèle de calcul où la composition de fonctions correspond à l'intérieur d'un agent et les envois de messages aux liens entre les agents potentiellement composés dans une organisation arborescente.

nouveauté. Dans l'ensemble du modèle nous avons une grande distinction entre agent/mémoire/séquence et multi/messages/graphe orienté acyclique.

Le second point sur lequel nous souhaitons insister est le fait que ce modèle multi-agent est paramétrable par un modèle de calcul pour l'agent. Ce point est intéressant pour envisager l'interaction entre agents écrits dans différents langages ou encore pour des agents utilisant différents modèles de calculs comme des modèles quantiques ou encore analogiques[2].

7.3.2 Perspectives

Déterminisme Nous avons vu que notre modèle ne recouvre pas uniquement des systèmes déterministes. La possibilité, sans changer la description des agents, de restaurer une exécution déterministe permettrait d'évaluer l'impact de cette source de non déterminisme et de le dissocier de l'impact des générateurs peudo-aléatoire dans nos expériences. Nous décrivons une première piste : tout d'abord nous introduirions un temps discret sur lequel l'ensemble du système serait synchronisé, à chaque pas de temps un agent ne pourrait traiter qu'un seul message, ensuite l'ensemble des messages produits par les traitements du pas de temps seraient ajoutés aux files d'agents en suivant un ordre total fixé à l'avance sur l'ensemble des agents.

Plate-forme de soutien Le code d'accompagnement de cette thèse a été construit sur un middleware de communication agent (CoRe DMS) qui ne reflète que peu le modèle que nous avons défini. Nous avons déjà commencé à mettre en place une plate-forme correspondant à notre modèle. Nous souhaitons également migrer nos agents sur cette plate-forme afin de pouvoir simplifier l'expression de nos agents dont le code mélange jusqu'ici des éléments spécifiques à notre système de recherche et des méthodes correspondant au modèle.

[2]Il faudrait alors faire attention à des hypothèses implicites du modèle comme la capacité à obtenir des copies d'un message dans le cadre d'un envoi aux fils qui contredirait le "No cloning theorem" quantique [WZ82].

Annexe A

Citations

Nous donnons ici des phrases courtes dans lesquelles différents auteurs donnent leur définition d'émergence. Il ne s'agit pas d'un état de l'art mais plutôt d'un aperçu de l'émergence par petites touches. Nous pensons que la juxtaposition de ces phrases permet un aperçu éclairant[1]. Les citations sont classées par ordre chronologique.

John Stuart Mill

All organized bodies are composed of parts similar to those composing inorganic nature, and which have even themselves existed in an inorganic state ; but the phenomena of life, which result from the juxtaposition of those parts in a certain manner, bear no analogy to any of the effects which would be produced by the action of the component substances considered as mere physical agents. To whatever degree we might imagine our knowledge of the properties of the several ingredients of a living body to be extended and perfected, it is certain that no mere summing up of the separate actions of those elements will ever amount to the action of the living body itself.

John Stuart Mill dans [Mil43, page 407]

[1] Ce que nous aurions appelé *insight* en anglais.

George Henry Lewes

Thus, although each effect is the resultant of its components, the product of its factors, we cannot always trace the steps of the process, so as to see in the product the mode of operation of each factor. In this latter case, I propose to call the effect an emergent. It arises out of the combined agencies, but in a form which does not display the agents in action. Galileo established the luminous principle of the independence of motions. This we may generalize as the independence of causal agents. Each agent, indestructible and independent, has its own individual value ; and the effect or combination of agents has two modes : in the one case we have an addition or mixture in the other a combination with an emergent

George Henry Lewes dans [Lew75, page 214] page 214

Marvin Minsky

Gestalt :The unexpected emergence, from a complex system, of a phenomenon that had not seemed inherent in that system's separate parts. Such "emergent" or collective phenomena show that a "whole is more than the sum of its parts". However, further research usually shows that such phenomena can be explained completely, once we also take into account the interactions among those parts as well as the peculiarities and deficiencies in the observer's own perceptions and expectations. There do not seem to be any important principles common to the phenomena that have been considered, from time to time, to be emergent beyond the contemporary inability to understand them. Thus, holistic views tend to become scientific handicaps when they undermine our determination to extend the boundaries of our comprehension.

Marvin Minsky dans [Min88, page 328]

Stephanie Forrest

The requirements for emergent computation are [...] (1) A collection of agents, each following explicit instructions (2) Interactions among the agents (according to the instructions), which form implicit global patterns at the macroscopic level i.e. epiphenomena (3) A natural interpretation of the epiphenomena as computations.

Stephanie Forrest dans [For90, page 2]

Alexis Drogoul

L'émergence est un concept flou, mal défini et souvent décrié. Il dénote d'une manière générale, dans les systèmes qualifiés de complexes ou de non-linéaires, l'apparition d'un effet global qui n'est pas déductible de la connaissance des causes locales. Dans un système multi-agent, il dénote donc l'apparition d'un phénomène global qui n'est pas explicitement programmée dans le comportement des agents.

Alexis Drogoul dans [Dro93, page 24]

James Crutchfield

three notions will be distinguished :

1. The intuitive definition of emergence : "something new appears" ;
2. Pattern formation : an observer identifies "organization" in a dynamical system ; and
3. Intrinsic emergence : the system itself capitalizes on patterns that appear.

James Crutchfield dans [Cru94, page 4]

Vince Darley

A true emergent phenomenon is one for which the optimal means of prediction is simulation.

Vince Darley dans [Dar94, page 2]

Timothy O'Connor

"novel causal influence". This term is intended to capture a very strong sense in which an emergent's causal influence is irreducible to that of the micro-properties on which it supervenes : it bears its influence in a direct, "downward" fashion, in contrast to the operation of a simple structural macro-property, whose causal influence occurs via the activity of the micro-properties which constitute it.

Timothy O'Connor dans [O'C94, page 15]

John Holland

The hallmark of emergence is this sense of much coming from little. This feature also makes emergence a mysterious, almost paradoxical, phenomenon smacking of "get rich quick" schemes.

John Holland dans [Hol97, page 2]

William Wimsatt

As science progress, emergence claims will be seen as nothing more than temporary confessions of ignorance.

William Wimsatt dans [Wim97, page 1]

Bonabeau

L'auto-organisation caractérise tout processus au cours duquel des structures émergent au niveau collectif [...] à partir de la multitude des interactions entre individus, sans être codées explicitement au niveau des individus.

Bonabeau dans [BT97]

Valérie Camps

Les notions d'émergence et d'auto-organisation sont fortement liées. L'auto-organisation introduit des comportements, des structures, ou des formes au niveau collectif qui sont nouveaux relativement aux comportements, structures ou formes des parties qui le composent : c'est l'émergence.

Valérie Camps dans [Cam98, page 76]

Ronald Arkin

Emergence is often invoked in an almost mystical sense regarding the capabilities of behavior-based systems. Emergent behavior implies a holistic capability where the sum is considerably greater than its parts.

Ronald Arkin dans [Ark98, page 105]

Jeffrey Goldstein

Emergent phenomena are conceptualized as occurring on the macro level, in contrast to the micro-level components and processes out of which they arise.

Jeffrey Goldstein dans [Gol99]

Nigel Gilbert

Emergence is an essential characteristic of social simulation. Indeed, without emergence, it might be argued that a simulation is not a social simulation.

Nigel Gilbert dans [Gil02]

Mark Bedau

There are a variety of notions of emergence, and they are contested. We can provide some order to this controversy by distinguishing two hallmarks of how macro-level emergent phenomena are related to their micro-level bases : emergent phenomena are dependent on underlying processes, emergent phenomena are autonomous from underlying processes.

Mark Bedau dans [Bed03, page 2]

Jean-Pierre Muller

Unfortunately, the notion of emergence itself is problematic for the modeller as well as for the designer, being most often defined by an absence of "something" as composability, predictability, and so on when the notion itself is not criticised. [...] A phenomenon is emergent if and only if we have :

- a system of entities in interaction whose expression of the states and dynamics is made in an ontology or theory D ;
- the production of a phenomenon, which could be a process, a stable state, or an invariant, which is necessarily global regarding the system of entities ;
- the interpretation of this global phenomenon either by an observer or by the entities themselves via an inscription mechanism in another ontology or theory D'.

Jean-Pierre Muller dans [Mul03]

Tom De Wolf et Tom Holvoet

A system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel with respect to the individual parts of the system.

Tom De Wolf et Tom Holvoet dans [DH05]

Peter Cariani

Emergence is the creative process by which new structures and functions come into being. There are two fundamental, complementary conceptions of emergence : combinatoric emergence, wherein novelty arises by new combinations of pre-existing primitives, and creative emergence, wherein novelty arises by de novo creation of new kinds of primitives.

Peter Cariani dans [Car06]

Annexe B

Glossaire

Le glossaire est organisé en trois parties. La première concerne les termes relatifs à l'émergence, la deuxième les notions de notre modèle multi-agent et la dernière concerne le domaine de la recherche heuristique.

B.1 Emergence

Causalité descendante : La causalité descendante ou *downward causation* désigne le fait qu'un émergent puisse exercer une causalité sur le système dont il émerge. Exercer une causalité signifie que l'émergent peut être la cause dans une relation de cause à effet.

Complexité de Kolmogorov : Ou complexité de Kolmogorov-Solomonoff-Chaitin ou complexité algorithmique. Correspond à la plus grande concision pour décrire un objet dans un formalisme Turing complet. Il s'agit donc de la longueur du plus court programme produisant l'objet.

Complexité statistique : Famille de mesures de complexité définie pour lesquelles une séquence aléatoire et une séquence constante ont toutes deux une complexité nulle. La complexité est alors distinguée du caractère aléatoire, contrairement au cas de la complexité de Komogorov qui est maximale pour des séquences aléatoires.

Emergence : L'objet principal de cette thèse. Désigne l'apparition de propriétés ou d'entités se situant à un niveau distinct de ce qui l'a produit ou découvert. Pour nous, une entité concernée par une synergie.

Epiphénomène : Phénomène qui n'exerce pas d'influence sur ce qui l'a produit. L'exemple courant est l'ombre d'un objet.

Survenance : Fait du survenir, désigne le fait qu'un émergent survient sur un niveau de base.

Synergie : Dynamique d'un système composé où les effets sont supérieurs à ceux obtenus quand les composants sont isolés.

B.2 Modèle multi-agent

Agent atomique : Agent constituant une feuille de l'arbre de l'organisation.

Agent composite : Agent constitué d'un sous-arbre de l'arbre d'organisation.

Classe d'agent : Catégorie ou sorte d'agent définie par un type de mémoire et des fonctions de traitements des messages.

Concurrent : Deux agents sont concurrents s'ils s'exécutent simultanément tout en échangeant de l'information.

Interaction : Envoi de message entre deux agents reliés par l'organisation du système.

Organisation : Arbre reliant les agents entre eux et conditionnant les interactions.

Parallèle : Deux agents s'exécutent en parallèle s'ils s'exécutent simultanément mais n'échangent pas d'information.

B.3 Recherche Heuristique

Comportement : Un comportement est une liste de couples décrivant une perception et l'action correspondante.

Evaluation : Fonction associant une valeur à un comportement.

Générateur : Fonction associant à un point des points du même espace.

Heuristique : (du grec heuriskein signifiant trouver) Méthode d'exploration d'un espace de recherche.

Heuristique itérative : Heuristique fondée sur l'itération d'un pas de recherche.

Métaheuristique : Heuristique utilisant une ou plusieurs autres heuristiques.

Pas de recherche : Elément d'une recherche effectuée par une heuristique itérative.

Point : Un point de l'espace de recherche.

Problème : Un problème est composé d'un espace de recherche, de générateurs et d'un testeur.

Simulation : Fonction permettant d'associer un comportement à un point de l'espace de recherche. Une simulation peut être paramétrée par un scénario.

Testeur : Fonction associant une valeur à un point par l'application d'une simulation et d'une évaluation.

Testeur principal : Désigne le testeur correspondant à l'objectif principal du système.

Testeur d'imitation : Testeur consistant à imiter un comportement donné. L'heuristique cherche à maximiser le nombre de cas où une solution se comporte comme le comportement fourni. Similaire à l'apprentissage supervisé.

Valeur : La valeur d'un comportement est le résultat de l'application d'une fonction d'évaluation sur ce comportement.

Voisinage : Le voisinage d'un point selon un générateur est le résultat de l'application (l'image) de ce générateur sur ce point ; il s'agit d'une partie de l'espace de recherche.

Annexe C

Notations algorithmiques

Cette annexe contient un court ensemble de notations pseudo algorithmiques utilisées dans le manuscrit.

Expressions, valeurs, types et noms

Nous manipulons principalement des expressions auxquelles sont associées des valeurs, ces valeurs appartenant à des types. Une expression peut être associée à un nom.

Les types primitifs sont les entiers de type Entier et les flottants de type Flottant.

Tuples, séquences, ensembles, alternatives

Tuples On notera un tuple ou n-uplet $< x_1, ..., x_n >$, une séquence est notée $[x_1, ..., x_n]$ et un ensemble $\{x_1, ..., x_n\}$. La séquence vide est donc [] et l'ensemble vide {}. Pour un type donné $Type$ on notera $[Type]$ le type des séquences d'éléments de $Type$ et $\{Type\}$ le type des sous-ensembles de l'ensemble des valeurs de $Type$. La notation ⟨ Type1,Type2 ⟩ correspond au type des couples ⟨ x1,x2 ⟩ tels que $x_i \in Type_i$. Une extension habituelle est la notion d'enregistrement soit un tuple dont les éléments sont associés à des noms, aussi on peut écrire le type suivant ⟨ nom1 :Type1,nom2 :Type2 ⟩ .

Nous définissons également l'opérateur *avec* qui permet de produire un nouvel enregistrement en n'en modifiant que certains éléments. Ainsi si on a un enregistrement x de valeur ⟨ numerateur=4,denominateur=5

$\rangle$ alors l'expression "x *avec numerateur* = 7" vaudra $\langle$ numerateur=7,denominateur=5 $\rangle$.

Afin de faciliter l'accès à un élément d'un tuple, nous avons permis d'y associer un nom. Ainsi, pour un type de tuples *rationnel* défini comme $\langle$ num :Entier,dénominateur :Entier $\rangle$ et une instance x égale à $\langle$ numerateur=7,denominateur=12 $\rangle$, on peut écrire $x.numerateur$ qui vaut alors 7.

Séquences Les opérations pour les séquences sont :

$vide(s)$ indique la séquence vide

$premier(s)$ le premier élément d'une séquence

$fin(s)$ la séquence sans son premier

$contient(s, element)$ indique si la séquence contient *element*

$succ(s, element)$ le successeur d'*element* dans la séquence, le premier si *element* est à la fin de la séquence

$ajoutdroite(s, element)$ ajout d'un élément à la fin de la séquence

$ajoutdroite(s, element, limite)$ ajout d'un élément à la fin de la séquence en supprimant le premier si la taille dépasse *limite*

Ensembles La présence d'ensemble peut poser problème ici, du fait de l'indéterminisme que cela peut provoquer. En effet, un opérateur simple tel que l'extraction d'un élément de l'ensemble suppose un choix non déterministe puisque rien indique quel élément doit être extrait. Les opérateurs sont :

$vide(e)$ indique l'ensemble vide

$taille(e)$ indique la taille de l'ensemble

$extrait(e, alea)$ extrait un élément de l'ensemble tiré au hasard avec la source *alea*

$meilleur(e)$ pour un ensemble muni d'un ordre, donne son élément

$meilleurs(e, n)$ pour un ensemble muni d'un ordre, donne les n meilleurs éléments

$union$ l'union de deux ensembles

$difference$ le premier ensemble privé des éléments du second

Résumé sur la partie fonctionnelle

Expression	Valeur	Type
6	6	Entier
6 * 1.0	6.0	Flottant
id [6,10,4]	[6,10,4]	[Entier]
premier [6,10,4]	6	Entier
reste [6,10,4]	[10,4]	[Entier]
v= ⟨ x=4,y=3 ⟩	⟨ x=4,y=3 ⟩	⟨ x :Entier,y :Entier ⟩
(v avec y=5)	⟨ x=4,y=5 ⟩	⟨ x :Entier,y :Entier ⟩

Spécificités agent

Nous donnons quelques spécificités nécessaires à la gestion d'agent et de messages.

Construction de messages Pour chaque type de message Xxx, nous spécifions une fonction Xxx prenant en paramètre un destinataire appartenant à $\{pere, fils, tous, boucle\}$ et sa valeur. Le constructeur associé à chaque nom de type permet alors de distinguer deux messages de même structure.

Type de messages Un agent qui traite un message peut accéder à son type par la fonction *type* : si m est un message de type t alors $type(m)$ vaut t.

Création d'agent Afin de modéliser la possibilité de créer un agent, un type de message particulier existe : $Creer(Type, Initiale)$ où type est le nom du type d'agent que l'on souhaite créer et Initiale est la valeur initiale de sa mémoire. L'agent est alors créé comme fils de l'expéditeur du message. Ce message est sans destinataire.

Annexe D

Règles du jeu du backgammon

Le jeu de backgammon est un jeu assez répandu dont les règles ne sont pas toujours complètes. Nous exposons ici l'ensemble des règles que nous avons utilisées dans notre prototype.

Le tableau est découpé en 4 jans. Chaque jan contient 6 positions ou *flèches*. L'état de départ est donné en figure (cf. figure D.1).

Nous donnons un petit peu de jargon :

bar/barre lieu où on trouve les pions frappés en attente d'être sortis.

blot/solitaire un pion adversaire isolé en sa position

bearing off/sortie position où l'on retire ses pions du plateau de jeu quand tous les pions du joueur sont dans son jan

frapper se dit quand un déplacement arrive sur un blot adverse qui se retrouve alors sur la barre

Sauts

A chaque tour, un joueur lance ses deux dés, les deux nombres produits permettent de déduire les saut possibles. Si les deux dés ont la même valeur, le joueur dispose de 4 sauts de cette valeur[1], si les valeurs sont différentes le joueur dispose d'un saut pour chaque valeur[2].

[1] 5 et 5 donne 4 sauts de 5 (5,5,5,5)

[2] 5 et 2 donne un saut de 5 et un saut de 2 (5,2)

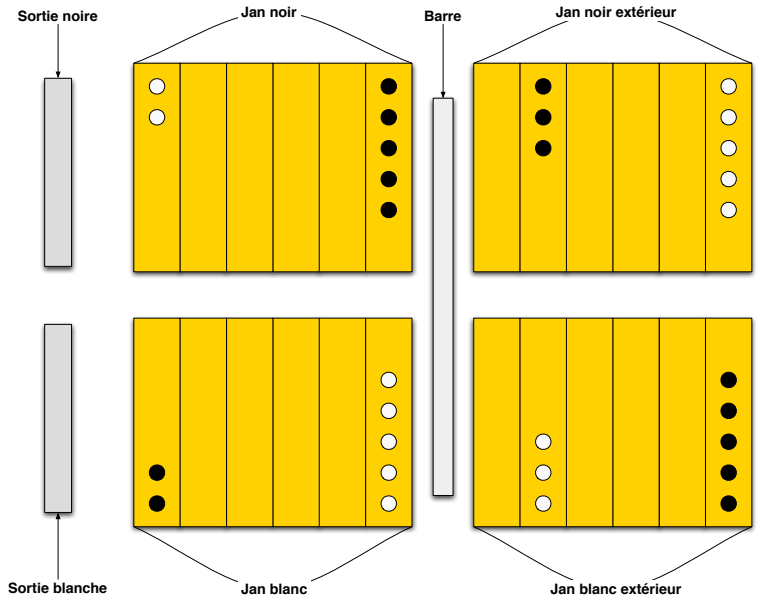

FIG. D.1 – Le plateau au début de la partie

Déplacement

Un déplacement se fait d'une flèche a à une flèche b. Nous détaillons maintenant les conditions sur a et b :

- a doit contenir un pion de la couleur du joueur
- a doit être la barre si le joueur a un pion sur la barre (on ne peut jouer d'autre pion, tant qu'il y en a sur la barre)
- b ne doit pas contenir plus d'un pion adverse

Cours normal d'une partie

Le jeu suit son cours normal tant que le joueur n'a pas rapatrié tous ses pions dans son jan. Dans ce cas :

- a et b doivent être distantes de la valeur d'un saut utilisable
- b ne peut être la sortie du joueur

Phase de sortie

La phase de sortie correspond au cas où le joueur peut retirer des pions c'est à dire les placer sur sa sortie. Dans ce cas :

- a et b doivent être distantes d'une valeur qui correspond à un saut utilisable
- sinon b doit être la sortie du joueur, a doit être la flèche la plus éloignée de la sortie et il faut qu'il existe un saut utilisable dont la valeur dépasse la distance entre a et b.

Cas de blocages

Le joueur doit jouer le coup permettant d'utiliser le maximum de sauts. Plusieurs cas sont possibles :

- aucun déplacement n'est possible, le joueur passe son tour
- on peut utiliser tous les sauts, le joueur doit les utiliser
- les deux sauts ne sont pas utilisables, le joueur doit choisir parmi les coups en utilisant le plus grand saut ; si c'est impossible, il doit jouer le petit saut
- un cas spécial où les deux sauts sont utilisables mais la configuration résultat rend impossible l'utilisation de l'autre saut (deux coups n'utilisant qu'un saut sur les deux) : le joueur doit utiliser le plus grand saut et non le plus petit (cf. figure D.2).

Résultat d'un déplacement

Le résultat d'un déplacement de a à b est le transfert d'un pion du joueur de la position a sur la position b. Si la position b contenait exactement un pion adverse, ce pion se retrouve sur la barre.

Gain

Une partie est gagnée dès qu'un joueur a tous ses pions sur sa sortie.

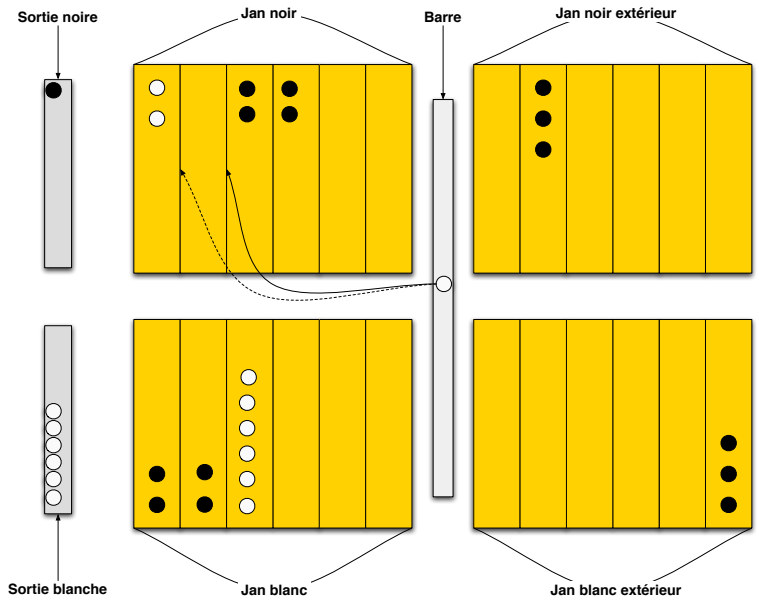

FIG. D.2 – Le pion blanc peut sortir en utilisant le déplacement de 1 ou 2. Comme il n'y a pas d'autre coup possible, le joueur doit utiliser le déplacement de 2.

Bibliographie

[Abb06] R. Abbott. Emergence explained. *Complexity*, 12(1) :pages 13–26, 2006.

[Aek99] F. Van Aeken. *Les Systèmes Multi-Agents Minimaux*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 1999.

[AHP05] F. Armetta, S. Hassas, and S. Pimont. A new protocol to share critical resources by self-organized coordination. In *Engineering Self-Organising Systems*, pages 90 – 103, Utrecht, Netherlands, 2005.

[Ang94] P.J. Angeline. *Advances in Genetic Programming*, chapter Genetic Programming and Emergent Intelligence, pages 75–98. MIT Press, 1994.

[Ark98] R.C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, 1998.

[Arm03] J. Armstrong. *Making reliable distributed systems in the presence of software errors*. PhD thesis, The Royal Institute of Technology, Stockholm, Sweden, 2003.

[BCK05] A. Le Bouthilier, T. Crainic, and P. Kropf. A guided cooperative search for the vehicle routing problem with time windows. *IEEE Intelligent Systems*, 20(4) :pages 36–42, 2005.

[BD97] E. Bonabeau and J.L. Dessalles. Detection and emergence. *Intellectica*, 2(25) :pages 85–94, 1997.

[Bed97] M.A. Bedau. Weak emergence. *Philosophical Perspectives : Mind, Causation and World*, 11 :pages 375–399, 1997.

[Bed03] M.A. Bedau. Downward causation and the autonomy of weak emergence. *Principia*, 6 :pages 5–50, 2003.

[Ben88] C.H. Bennett. *The Universal Turing Machine, a Half-Century Survey*, chapter Logical Depth and Physical Complexity, pages 227–257. Oxford University Press, 1988.

[BG07] F. Boschetti and R. Gray. *Advances in Applied Self-organizing Systems*, chapter A Turing Test for Emergence. Springer Verlag, 2007.

[BM97] N. Bensaid and P. Mathieu. A hybrid and hierarchical multi-agent architecture model. In *PAAM*, pages pages 145–155, 1997.

[BSF03] G. Beurier, O. Simonin, and J. Ferber. Un modèle de système multi-agents pour l'émergence multi-niveaux. *Technique et Science Informatiques*, 22(4) :pages 235–247, 2003.

[BT97] E. Bonabeau and G. Theraulaz. Auto-organisation et comportements collectifs : la modélisation des sociétés d'insectes. In *Auto-organisation et comportement*. Hermès, 1997.

[Bun77] M.A. Bunge. Emergence and the mind. *Neuroscience*, 2(4) :pages 501–509, 1977.

[Cam98] V. Camps. *Vers une théorie de l'auto-organisation dans les systèmes multi-agents basée sur la coopération : application à la recherche d'information dans un système d'information répartie*. PhD thesis, Université Paul Sabatier, Toulouse, France, 1998.

[Car91] P. Cariani. Emergence and artificial life. In *Artificial Life 2*, pages pages 775–798, Sante Fe, USA, 1991.

[Car06] P. Cariani. On creating new informational primitives in minds and machines (http ://homepage.mac.com/cariani/carianiwebsite/carianitempleton2006.pdf). 2006.

[CDS89] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal brain damage. In *NIPS*, pages 598–605, Denver, USA, 1989.

[CF02] M. Christen and L.R. Franklin. The concept of emergence in complexity science : Finding coherence between theory and practice. In *Proceedings of the Complex Systems Summer School*, 2002.

[CG98] L. Cardelli and A. Gordon. Mobile ambients. In *Lecture Notes in Computer Science*, volume 1378, pages 253–292, 1998.

[Che04] V. Chevrier. From self-organized systems to collective problem solving. In *ESAW*, pages 222–230, Toulouse, France, 2004.

[CMK02] A. Cornuéjols, L. Miclet, and Y Kodratoff. *Apprentissage artificiel*. Eyrolles, 2002.

[Cor02] P.A. Corning. The re-emergence of "emergence". *Complexity*, 7(6) :pages 18–30, 2002.

[Cru94] J.P. Crutchfield. The calculi of emergence : Computation, dynamics, and induction. *Physica D*, 75 :pages 11–54, 1994.

[Dar94] V. Darley. Emergent phenomena and complexity. In R. Brooks and P. Maes, editors, *Artificial Life 4*, pages 411–416, 1994.

[DD05] J. Deguet and Y. Demazeau. A complexity based feature to support emergence in mas. In *Proceedings of the International Central and Eastern European Conference on Multi-Agent Systems*, pages 616 – 619, Budapest, Hungary, September 2005.

[DDM06] J. Deguet, Y. Demazeau, and L. Magnin. Elements about the emergence issue, a survey of emergence definitions. *ComPlexUs, International Journal on Modelling in Systems Biology, Social, Cognitive and Information Sciences*, 3(1) :pages 24–31, 2006.

[Deg04] J. Deguet. L'émergence dans les systèmes multi-agents. Master's thesis, Université Joseph Fourier, 2004.

[Dem95] Y. Demazeau. From interactions to collective behaviour in agent-based systems. In *Proceedings of the First European conference on cognitive science*, pages 117–132, Saint Malo, France, 1995.

[Dem01] Y. Demazeau. *VOYELLES, Habilitation à Diriger des Recherches*. PhD thesis, Institut National Polytechnique de Grenoble, 2001.

[DFP95] J. Dassow, R. Freund, and G. Paun. Cooperating array grammar systems. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(6) :pages 1029–1053, 1995.

[DH05] T. DeWolf and T. Holvoet. Towards a methodolgy for engineering self-organising emergent systems. In *SOAS*, pages 18–34, Glasgow, UK, 2005.

[DM91] Y. Demazeau and J.P. Muller, editors. *From reactive to intentional agents*. Elsevier, 1991.

[Dro93] A. Drogoul. *De La Simulation Multi-Agent A La Résolution Collective de Problèmes*. PhD thesis, Université Paris VI, Paris, France, 1993.

[DS04] M. Dorigo and T. Stutzle. *Ant Colony Optimization*. MIT Press, 2004.

[Erl] Erlang programming language, http ://www.erlang.org.

[Fer95] J. Ferber. *Les systèmes multi-agents. Vers une intelligence collective.* InterEditions, 1995.

[FF96] H. Fernau and R. Freund. Bounded parallelism in array grammars used for character recognition. In *6th International Workshop on Advances in Structural and Syntactical Pattern Recognition*, pages 40–49, Leipzig, Germany, 1996.

[For90] S. Forrest. *Emergent Computation.* MIT Press, 1990.

[Gaf95] F. Gaffiot. *Dictionnaire latin français.* Hachette, 1995.

[Gar70] M. Gardner. Mathematical games : The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223 :pages 120–123, 1970.

[Gil02] N. Gilbert. Varieties of emergence. In *Agent 2002 Conference : Social agents : ecology, exchange, and evolution*, Chicago, USA, 2002.

[Glo89] F. Glover. Tabu search–part 1. *ORSA Journal On Computing*, 1(3) :pages 190–206, 1989.

[GM97] F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5 :pages 317–342, 1997.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[Gol99] J. Goldstein. Emergence as a construct : History and issues. *Emergence*, 1(1) :pages 49–72, 1999.

[HBS73] C. Hewitt, P. Bishop, and R. Steiger. A universal modular actor formalism for artificial intelligence. In *IJCAI*, pages 235–245, Stanford, USA, 1973.

[Hew06] C. Hewitt. What is commitment ? physical, organizational, and social. In *COIN (AAMAS workshop)*, Hakodate, Japan, 2006.

[HM97] P. Hansen and N. Mladenovic. *Handbook of Metaheuristics*, volume 57, chapter Variable Neighborhood Search, pages 145–184. Springer, 1997.

[Hol75] J.H. Holland. *Adaptation in natural and artificial systems.* MIT Press, 1975.

[Hol97] J.H. Holland. *Emergence : From Chaos to Order.* Perseus Books, New York, 1997.

[Hop82] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the*

National Academy of Sciences of the United States of America, 79(8) :pages 2554–2558, 1982.

[HW07] J. Halley and D. Winkler. Classification of emergence and its relation to self-organization. *Complexity*, 2007.

[Jyt] Jython project, http ://www.jython.org/.

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :pages 671–680, 1983.

[Kim99] J. Kim. Making sense of emergence. *Philosophical Studies*, 95 :pages 3–36, 1999.

[Koz92] J. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992.

[Kub01] A. Kubik. On emergence in evolutionary multiagent systems. In *ECAL*, pages 326–337, Prague, 2001.

[Kub03] A. Kubik. Toward a formalization of emergence. *Artificial Life*, 9 :pages 41–66, 2003.

[Lam78] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7) :pages 558–565, 1978.

[Lan90] C.G. Langton. Computation at the edge of the chaos, phase transition and emergent computation. *Physica D Nonlinear Phenomena*, 42(1) :pages 12–37, 1990.

[Lar96] *Le petit Larousse illustré*. Larousse, 1996.

[Lew75] G.H. Lewes. *Problems of Life and Mind, The foundations of a creed.* Trubner and Company, 1875.

[LP02] S. Luke and L. Panait. Lexicographic parsimony pressure. In *GECCO 2002*, pages 829–836, New York, USA, 2002.

[Mag] Projet magique, http ://www2.lifl.fr/smac/projects/magique/.

[McK05] E. McKean, editor. *The New Oxford American Dictionary, Second Edition*. Oxford University Press, 2005.

[Mil43] J.S. Mill. *System of Logic*. John W. Parker, 1843.

[Min88] M. Minsky. *The Society of Mind*. Simon and Schuster, 1988.

[MR04] M. Milano and A. Roli. Magma : A multiagent architecture for metaheuristics. *IEEE Transactions on systems man and cybernetics*, 34(2) :pages 925–941, 2004.

[Mul03] J.P. Muller. Emergence of collective behaviour and problem solving. In *ESAW*, pages 1–21, London, UK, 2003.

[O'C94] T. O'Connor. Emergent properties. *American Philosophical Quaterly*, 31 :pages 91–104, 1994.

[O'C00] T. O'Connor. Causality, mind, and free will. *Philosophical Perspectives*, 34 :pages 105–117, 2000.

[OL96] I.H. Osman and G. Laporte. Metaheuristics : A bibliography. *Annals of Operations Research*, 63(5) :pages 511–623, 1996.

[ØL03] E.H. Østergaard and H.H. Lund. Evolving control for modular robotic units. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation*, Kobe, Japan, 2003.

[Qui00] J. Quinqueton. Emergence et sémantique des mondes possibles pour des agents appris. In *JFIADSMA 2000*, St Etienne, France, 2000.

[Rab05] N. Rabinowitz. Emergence : an algorithmic formulation. Master's thesis, University of Western Australia, 2005.

[RS01] E. Ronald and M. Sipper. Surprise versus unsurprise : Implications of emergence in robotics. *Robotics and Autonomous Systems*, 37(1) :pages 19–24, 2001.

[RSC99] E. Ronald, M. Sipper, and M.S. Capcarrère. Design, observation, surprise ! a test of emergence. In *Artificial Life 5*, pages 225–239, 1999.

[Saw00] R.K. Sawyer. Simulating emergence and downward causation in small groups. In *MABS*, pages 49–67, Boston, USA, 2000.

[Sea92] J.R. Searle. *The Rediscovery of the Mind*. Cambridge, 1992.

[Sha] C. Shalizi. Notebook on emergent properties (http ://www.cscs.umich.edu/ crshalizi/notebooks/emergent-properties.html).

[Sha01] C. Shalizi. *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. PhD thesis, University of Wisconsin, Madison, USA, 2001.

[Sta] Stanford encyclopedia : Emergent properties (http ://plato.stanford.edu/entries/properties-emergent/).

[Sta01] R.K. Standish. On complexity and emergence. *Complexity International*, 9, 2001.

[Ste95] L. Steels. The spontaneous self-organization of an adaptive language. In *Machine Intelligence 15*, pages 205 – 224, Oxford, England, 1995.

[Ste99] A. Stephan. *Emergenz : Von der Unvorhersagbarkeit zur Selbstorganisation*. Dresden University Press, 1999.

[tc97] M.R. Jean (travail collectif). Emergence et sma. In *JFIAD-SMA'97*, pages 323–342, La Colle-sur-Loup, France, 1997.

[Tes95] G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3) :pages 58–68, 1995.

[TLF] Trésor de la langue francaise informatisé, http ://atilf.atilf.fr/.

[Tur50] A.M. Turing. Computing machinery and intelligence. *Mind*, 49 :pages 433–460, 1950.

[Wim97] W. Wimsatt. Aggregativity : Reductive heuristics for finding emergence. *Philosophy of Science*, 64(4) :pages 372–384, 1997.

[WM97] D. Wolpert and W. Macready. No free lunch theorems for optimization. In *IEEE Transactions on Evolutionary Computation*, volume 1, pages 67–82, 1997.

[WP02] C. Webber and S. Pesty. Emergence de diagnostic par formation de coalitions - application au diagnostic des conceptions d'un apprenant. In *JFIADSMA*, Lille, France, 2002.

[WZ82] W.K. Wootters and W.H. Zurek. A single quantum cannot be cloned. *Nature*, 299 :pages 802 – 803, 1982.

[Yao99] X. Yao. Evolving artificial neural networks. In *Proceedings of IEEE*, volume 87, pages 1423–1447, 1999.

Abstract Emergence and multi-agent systems are two domains with share similar problems. Through the study of emergence in the context of multi-agent systems, this work is centred on their principal common point ; the search for a collective advantage gained through the interaction among the agents, when the global result is due to interaction. This situation is often summarized by a whole that is more than the sum of its parts. One contribution of this thesis is identifying interactions whose impact can be evaluated through the side-by-side comparison of systems including these interactions or not. Such collective benefits are defined for problem-solving by a heuristic search with a hierarchical multi-agent model. This work includes a multi-agent model of that kind of search that identifies collective benefits, which are called synergies. Three kinds of synergies are considered : the synergy between heuristics searching the same problem, the one between similar problems, and the synergy between a human user and the artificial search system. These possible synergies are included in the discussion about emergence. In this framework, some heuristic populations match the idea of a "true composite" that is a composed system whose result is due to interactions between components and not to its composition. The interaction between local and global levels fits in the levels induced by the model's hierarchical organisation. This way, a contribution to the study of emergence is given, by the definition possibilities that a multi-agent model allows for emergence.

Keywords emergence, heuristic search, synergy, multi-agent systems, interactions

Résumé L'émergence et les systèmes multi-agent sont deux domaines aux problématiques proches. A travers l'étude de l'émergence dans le cadre des systèmes multi-agent, notre travail consiste à envisager leur principal point commun : la recherche d'un avantage collectif gagné grâce aux interactions entre les agents, quand le résultat global du système qui découle de l'exécution des agents est attribuable à l'interaction. Cette situation est souvent décrite comme celle d'un tout supérieur à la somme de ses parties. Une contribution de cette thèse est d'identifier des interactions dont l'impact peut être évalué à travers la comparaison de systèmes utilisant ou non ces interactions. Nous définissons de tels gains collectifs pour la résolution de problèmes par recherche heuristique dans un modèle d'agents hiérarchique. Ce travail inclut une modélisation multi-agent de ce type de recherche permettant de mettre en évidence des gains collectifs que nous appelons synergies. Trois synergies sont envisagées : la synergie entre des heuristiques travaillant sur le même problème, celle entre des problèmes proches et la synergie entre un utilisateur et le système de recherche artificiel. Ces possibilités de synergies sont à replacer dans le cadre de la discussion concernant l'émergence. Dans ce cadre, certaines populations d'heuristiques correspondent à l'idée de "vrai composite" qui désigne des systèmes composés dont le résultat est attribuable aux interactions entre composants et non à leur composition. L'interaction entre niveaux globaux et locaux s'envisage également naturellement à travers les niveaux induits par l'organisation hiérarchique de notre modèle. C'est en ce sens que le travail fourni contribue à l'étude de l'émergence : par l'étude des possibilités de définition offertes par un modèle multi-agent.

Mots Clés émergence, recherche heursitique, synergies, systèmes multi-agent, interactions