Autour du problème du consensus

Clément Pira, Amal El Fallah-Seghrouchni

Autour du problème du consensus

Clément PIRA Amal El Fallah Seghrouchni
Clement.Pira@lip6.fr Amal.ElFallah@lip6.fr

Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie
104, avenue du Président Kennedy
75016 Paris – FRANCE

Résumé :
Dans ce papier, le problème d'atteinte de consensus est étudié relativement à trois domaines : la décision collective, la théorie des jeux et l'algorithmique répartie. Le premier domaine étudie les conditions générales d'existence d'un consensus (*i.e.* existence de fonctions d'agrégation). Les deux autres tentent d'en comprendre la dynamique. Il en ressort deux problématiques de l'implémentation : 1) pour l'algorithmique répartie, il s'agit de s'assurer de la diffusion suffisante de la connaissance au sein d'un système pouvant par exemple tolérer les fautes (répartition) ; 2) pour la théorie des jeux, il s'agit de trouver une correspondance entre "équilibre" stratégique et "optimum" social (compétition). De notre point de vue, le consensus multiagent réunit ces deux problèmes, d'où le besoin de développer un cadre commun aux deux domaines.

Mots-clés : Agent, consensus, répartition

Abstract:
In this paper, the consensus problem is studied through three fields : collective decision, game theory and distributed algorithmic. We identify two implementation problems : 1) in distributed algorithmic, we have to deal with communications and processes faults (distribution) ; 2) in game theory, we want to find an equilibrium which might be different from the optimal solution (competition). A model for multiagent system is presented as a compromise between models from game theory and distributed algorithmic.

Keywords: Agent, consensus, distribution

1 Introduction

Objectif. L'objectif de ce papier[1] est de proposer un modèle permettant d'étudier le problème du consensus dans les systèmes multiagents. Selon nous, les modèles opérationnels de SMA se trouvent à la croisée entre les modèles de théorie des jeux (permettant de capturer la notion de rationalité et de compétition) et les modèles d'algorithmique répartie (insistant plus sur les propriétés de cohérence et de terminaison, à travers l'étude des mécanismes tels que la communication ou la synchronisation). Nous tentons ainsi d'en présenter les points communs et les différences profondes. Dans le domaine multiagent, les modèles de théorie des jeux sont bien connus (les décisions au sein d'un SMA sont en effet élaborées par des agents possédant des intérêts individuels qu'il faut respecter lors du passage à la rationalité collective). Cependant, ils permettent surtout de donner une description de haut niveau du comportement d'un système. A plus bas niveau interviennent des problèmes étudiés par l'informatique répartie et généralement moins connus.

Problème de l'implémentation. L'approche "classique" de la décision collective fait intervenir la notion de fonction d'agrégation $\mathcal{F} : \mathcal{X}^{\mathcal{A}} \rightarrow \mathcal{X}$ et cherche par exemple à ramener cette fonctionnelle à une intégrale de Choquet (ou à un polynôme latticiel) pondérée par un jeu coalitionnel ν (un jeu simple). Ce jeu représente le poids donné aux coalitions d'agents. On démontre ainsi des théorèmes "à la Riesz" établissant des bijections entre des classes de jeux coalitionnels (structures décisionnelles) et des types de fonctions d'agrégations [4, 1] :

$$\nu \mapsto \left(x \mapsto (\oint_{\mathcal{A}}) x \, d\nu \right)$$

[1] Ce travail est financé par la DGA

A un niveau plus abstrait, il s'agit d'étudier la possibilité d'agréger des données en fonction de la topologie de l'espace sous-jacent [2]. Cependant, une fonction d'agrégation n'est pas une implémentation. Nous distinguons en particulier deux problématiques d'implémentation associées à la concurrence :

La distribution : Pour réaliser une procédure de décision dans un système réparti, on est confronté aux problèmes de la transmission imparfaite de l'information ou encore aux défaillances du système. La difficulté supplémentaire concerne ici la *distribution asynchrone*.

La compétition : En théorie des jeux, implémenter c'est réaliser un choix collectif pour lequel on connait une solution optimale par un ensemble d'agents individualistes (*mecanism design*). Dans ce cas, c'est la notion d'équilibre qui prévaut. Et elle ne correspond par à la notion d'optimum, d'où des situations paradoxales de type "dilemme du prisonnier". La difficulté, c'est la *compétition*.

La distribution et la compétition ont conduit à deux types de solutions répondant à deux grandes problématiques : celle de la cohérence globale *du système et celle de la* rationalité collective*. Dans les deux cas, il faut développer un modèle de la dynamique du système. Cependant un modèle de théorie des jeux insistera plus sur la représentation de la rationalité individuelle des agents tout en limitant la modélisation de l'environnement ou du système de communication tandis qu'un modèle d'informatique répartie prendra le point de vue inverse.*

Problème du consensus. L'informatique répartie étudie également des tâches de décision ; en particulier des tâches de consensus $T : \mathcal{X}^{\mathcal{A}} \to \mathbb{P}(\mathcal{X})$. Cependant il importe ici avant tout de garantir :

- *l'atteinte d'un accord* : tous les agents corrects décident la même valeur ;
- *la terminaison du processus* : tous les agents corrects finissent pas décider.

Le défaut de cette approche, lorsqu'on l'applique aux systèmes multiagents, est la faible rationalité imposée à cette tâche de consensus. Généralement on impose une condition d'unanimité (*si tous les processus sont d'accord initialement, alors leur valeur doit être choisie collectivement*), bien plus faible que celles classiquement imposées à une fonction d'agrégation en théorie de la décision.

La théorie de la décision et l'informatique répartie sont toutes deux confrontées à des résultats d'impossibilité. Notre but est ainsi de comprendre les compromis à faire entre cohérence, terminaison et rationalité, pour garantir l'existence de protocoles implémentant certaines tâches de décision.

2 Modèle proposé

Le modèle proposé repose sur différents composants. Certains d'entre eux, comme la représentation des stratégies, s'inspirent d'outils classiques en théorie des jeux, alors que d'autres s'inspirent des concepts issus de l'algorithmique répartie comme l'exécution asynchrone ou encore la communication.

Réaction du système. Soit Ω un ensemble de *configurations* (représentant les états du système dans sa globalité[2]) et $\mathbb{A}$ un ensemble d'actions pouvant s'y produire permettant ainsi de passer d'une configuration à une autre. Soit finalement $\omega_0 \in \Omega$ une configuration initiale et $\overset{\bullet}{\to}: \mathbb{A} \times \Omega \to \mathbb{P}(\Omega)$ une fonction de transition non déterministe entre configurations. La *réaction* du système dans sa globalité est ainsi décrite par un automate $\langle \Omega, \mathbb{A}, \overset{\bullet}{\to}, \omega_0 \rangle$, généralement

[2] Aucun n'agent n'aura accès à l'intégralité de cet état.

un arbre enraciné en ω_0 (arbre de décision ou arbre de synchronisation en fonction du domaine). Si $a \in \mathbb{A}$ est une action s'appliquant à une configuration ω conduisant à une configuration ω', on notera :

$$\omega \xrightarrow{a} \omega'$$

Actions simultanées ou alternées. Le modèle d'actions collectives le plus simple est celui des actions *simultanées*. Dans ce cas on définit $\mathbb{A} = \prod_{\alpha \in \mathcal{A}} \mathbb{A}_\alpha$ (une action globale est un vecteur d'actions individuelles). Le but pour un agent α est de définir une stratégie déterministe $s_\alpha : \Omega \to \mathbb{A}_\alpha$ (ou stochastique $s_\alpha : \Omega \to \pi(\mathbb{A}_\alpha)$). Ensuite, dans une configuration ω donnée, chaque agent déroule sa stratégie et propose ainsi une action. Le vecteur d'actions est ensuite appliqué au système qui réagit et produit une nouvelle configuration ω'.

$$\omega \xrightarrow{(s_{\alpha_1}(\omega),\cdots,s_{\alpha_n}(\omega))} \omega'$$

Un autre modèle est celui des actions *alternées* (et donc sérialisées). Dans ce cas on définit $\mathbb{A} = \sum_{\alpha \in \mathcal{A}} \mathbb{A}_\alpha$ (une action globale est un couple composé d'un agent et de l'action qu'il propose). Il faut alors décider qui peut prendre une décision et quand. Un modèle simple est d'associer un décideur par configuration, au moyen d'une fonction $\mathrm{ag} : \Omega \to \mathcal{A}$, ce qui revient à partitionner Ω en $(\Omega_\alpha)_{\alpha \in \mathcal{A}}$. Chaque agent doit alors proposer une stratégie $s_\alpha : \Omega_\alpha \to \mathbb{A}_\alpha$. Dans une configuration ω, l'agent qui a la main déroule sa stratégie. L'action ainsi produite s'applique au système qui réagit pour donner une nouvelle configuration.

$$\omega \xrightarrow{(\mathrm{ag}(\omega),s_{\mathrm{ag}(\omega)}(\omega))} \omega'$$

Ordonnancement généralisé.

En théorie des jeux, on parle de jeux synchrones et asynchrones pour désigner les modèles d'actions simultanées et alternées. Nous n'adopterons pas cette définition dans la mesure où l'on souhaite faire le lien avec les modèles d'algorithmique répartie. En effet, dans ce domaine, l'asynchronisme fait référence à l'absence de temps global, d'horloges ou de deadlines ; et dans ce sens les modèles de théorie des jeux sont toujours synchrones.

Pour définir une notion de jeu réellement asynchrone, on commence par généraliser la notion d'action. Entre les deux extrêmes donnés par les modèles d'actions simultanées et alternées, on peut définir un modèle pour lequel, dans une configuration donnée, un sous-ensemble des agents est autorisé à agir. Ils produisent donc une action partielle à valeur dans $\mathbb{A} = \mathbb{A}_{[\mathcal{A}]} = \prod_{\alpha \in \mathcal{A}} (\mathbb{A}_\alpha + \bot)$ ([3]). D'autre part, plutôt que de fixer les agents ayant le droit d'agir au niveau de chaque configuration, on définit séparément un ordonnancement comme une suite de sous-ensembles d'agents ayant le droit d'agir. On note $\Sigma = \mathbb{P}(\mathcal{A})^{\mathbb{N}}$ l'ensemble des ordonnancements (*schedules*).

Soit $g \in \Omega$ un but (une configuration désirée par les agents) et soit $S \in \mathbb{P}(\mathcal{A})$ un sous-ensemble d'agents. Si tout chemin depuis la configuration initiale ω_0 jusqu'à ce but g contient un sous-chemin d'une longueur fixée faisant intervenir simultanément tous les agents de S, cela exprime le fait que les agents de S doivent d'une certaine manière se synchroniser au moins une fois lors de l'exécution s'ils souhaitent atteindre ce but g ; et donc que **le problème n'a pas de solution en asynchrone**. En particulier, dans un système asynchrone, toute suite d'actions doit pouvoir être sérialisée. La synchronisation est

[3] $X + \bot$ désigne l'ensemble X augmenté d'un élément supplémentaire $\bot$.

une composante essentielle de la coordination. C'est même une condition minimale pour qu'un groupe d'agents puisse *agir comme un seul*. Généralement, elle est supposée acquise en théorie des jeux qui se focalise sur une coordination de plus haut niveau (coordination des intérêts des agents). Cependant, d'un point de vue pragmatique, ce type de coordination est impossible sans un accord préalable sur le temps (quand peut-on agir ? quand prend fin la décision ? etc.).

D'autre part, il s'avère également nécessaire de décrire plus finement le système. Classiquement en théorie des jeux, la réaction du système est décrite comme un tout par une relation de transition $\xrightarrow{\bullet}$ (une fonction τ) ; et les agents sont décrits principalement par leur stratégie comportementale s_α et par leur vision du système (fonction de projection π_α décrite plus loin). Ils n'ont pas d'états internes à partir desquels la configuration globale peut être calculée. Dans un modèle de système réparti, le point de vue est radicalement opposé : l'évolution d'un agent est décrite localement par un automate et celle du système par le produit de ces automates. Lorsqu'un agent peut communiquer avec un groupe d'agents de manière atomique, il peut être assuré du fait que les états de croyances de ces agents sont cohérents les uns par rapport aux autres. Inversement, dans le cas contraire où les communications se font entre deux agents, un agent malicieux peut profiter de cette propriété du système pour induire des états de croyances incohérents chez les autres agents (par exemple faire croire deux choses incompatibles à deux agents différents) : c'est le principe des *agents byzantins*. Ici, la question n'est donc plus, pour un ensemble d'agents, de produire simultanément une action, mais de *subir une même action comme un seul agent*.

Notre modèle est similaire aux modèles d'automate entrée/sortie [8] dans le sens où il permet de modéliser une communication impliquant un émetteur et un récepteur au moyen d'actions d'émission sur un canal. Cependant ces canaux nous servent ensuite à redéfinir finement la notion d'état puis de configuration du système. Finalement, le modèle d'exécution d'un agent se rapproche plus de la notion de stratégie comportementale développée en théorie des jeux.

Soit $\hat{\mathcal{A}}$ un ensemble de composants[4] et $\mathcal{C}$ un ensemble de canaux entre ces composants. On note respectivement $\mathcal{C}(\alpha, \beta)$ l'ensemble des canaux d'origine α et d'extrémité β, $\mathcal{C}(\alpha, \bullet)$ ceux d'extrémité quelconque ou encore $\mathcal{C}(\alpha, *)$ ceux d'extrémité autre que α (absence de boucle). De manière symétrique, on définit $\mathcal{C}(\bullet, \beta)$ et $\mathcal{C}(*, \beta)$.

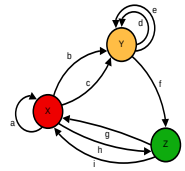


FIG. 1 – Trois composants $\{X, Y, Z\}$ reliés par neuf canaux $\{a, b, c, d, e, f, g, h, i\}$

A chaque canal $c \in \mathcal{C}$ est associé un type T_c représentant le type de données pouvant l'emprunter. On le complète en $T_c + \perp$ pour représenter l'éventualité d'un canal vide. Si C est un ensemble de canaux, on lui associe naturellement le type produit $T_{[C]} = \prod_{c \in C} (T_c + \perp)$. Le type $T_{[\mathcal{C}]}$ est assimilé à l'ensemble des actions globales $\mathbb{A}$. Ceci nous permet également de préciser la forme des actions que peut produire ou subir un agent α. On note ainsi $\mathbb{A}_{*,\alpha}$ (*resp.* $\mathbb{A}_{\alpha,*}$) le type d'entrée (*resp.* de sortie) de

[4] $\mathcal{A} \subseteq \hat{\mathcal{A}}$, par exemple $\hat{\mathcal{A}}$ = ensemble d'agents $\mathcal{A}$ + environnement ε.

l'agent α :

$$\mathbb{A}_{*,\alpha} = T_{[\mathcal{C}(*,\alpha)]} = \prod_{c \in \mathcal{C}(*,\alpha)} (T_c + \bot)$$

Finalement, on définit $\mathfrak{X}_\alpha = \mathbb{A}_{\alpha,\alpha}$, l'ensemble des valeurs des canaux bouclant sur un composant α. Cet ensemble peut être identifié à l'ensemble des états internes du composants[5]. Une *configuration* est quant à elle constituée des états locaux de chaque agent ainsi que de l'état du système de communication (mémoire ou système de messages) :

$$\Omega = \prod_{\alpha \in \hat{\mathcal{A}}} \mathfrak{X}_\alpha = \mathfrak{X}_\varepsilon \times \prod_{\alpha \in \mathcal{A}} \mathfrak{X}_\alpha$$

Le comportement d'un agent α est alors décrit par deux fonctions (stratégie comportementale et fonction de transition interne) laissant apparaître une symétrie entre une partie proactive et une partie réactive :

$$\left\{ \begin{array}{lcl} \varsigma_\alpha & : & \mathfrak{X}_\alpha \to \mathfrak{X}_\alpha \times \mathbb{A}_{\alpha,*} \\ \tau_\alpha & : & \mathbb{A}_{*,\alpha} \times \mathfrak{X}_\alpha \to \mathfrak{X}_\alpha \end{array} \right.$$

La *partie proactive* décrit la part du comportement de l'agent qu'il déclenche lorsqu'il est autorisé à agir par l'ordonnanceur. Soit $S \in \mathbb{P}(\hat{\mathcal{A}})$ est un ensemble d'agents auxquels l'ordonnanceur a attribué un pas de calcul. On définit le comportement proactif d'un agent $\alpha \in \mathcal{A}$ durant ce laps de temps par :

$$x \mapsto \left\{ \begin{array}{ll} \varsigma_\alpha(x) & \text{si } \alpha \in S \\ (x, (\bot, \cdots, \bot)) & \text{sinon} \end{array} \right.$$

La *partie réactive* décrit la part du comportement de l'agent activé en réaction à son contexte d'exécution (par exemple l'environnement). Si tous les canaux d'entrée d'un agent $\alpha \in \mathcal{A}$ sont vides, l'agent ne doit pas réagir d'où la contrainte :

$$\tau_\alpha((\bot, \cdots, \bot), x) = x$$

[5]Le fait de produire sur un canal ce que l'on récupérera au tour suivant constitue le principe d'une mémoire.

Finalement, on donne ci-dessous (figure 2) la formalisation d'une exécution asynchrone d'un ensemble d'agents décrits par leurs stratégies comportementales et leurs fonctions de transitions internes, le tout paramétré par un ordonnancement.

```
procedure execute(σ : Σ,
      ς : ∏_{α∈A} (X_α → A_{α,•}),
      τ : ∏_{α∈A} (A_{•,α} → X_α))
var x : A ;
begin
      forall [r ∈ N] do      // r = n° tour
      begin
          forall [α ∈ σ_r] do x[C(α,•)] ← ς_α(x[C(α,α)])
          forall [α ∈ A \ σ_r] do x[C(α,∗)] ← (⊥, ··· , ⊥)
          forall [α ∈ A] do x[C(α,α)] ← τ_α(x[C(•,α)])
      end
end
```

FIG. 2 – Exécution paramétrée par un ordonnancement σ

Cette construction permet d'isoler la contribution de l'ordonnanceur dans la décision collective. Chaque agent propose une stratégie et celui-ci propose un ordonnancement σ dans l'ensemble $\Sigma = \mathbb{P}(\mathcal{A} + \varepsilon)^{\mathbb{N}}$. On peut alors étudier le problème de la décision en fonction de contraintes faites sur cet ensemble : restriction à des ordonnancements plus ou moins synchrones, ajout d'une mesure de probabilité indiquant la vraisemblance d'apparition des ordonnancements, etc.

Observation partielle. De nombreux résultats en informatique répartie repose sur le fait que chaque agent n'a qu'une vision partielle du monde et que certaines configurations sont donc indistinguables de son point de vue. On définit ainsi $\mathfrak{X}_\alpha$ comme le type de données que perçoit l'agent α de l'ensemble des configurations Ω. On le dote également d'une fonction de projection $\pi_\alpha : \Omega \to \mathfrak{X}_\alpha$. On le restreint finalement à utiliser une stratégie s_α *uniforme*, c'est-à-dire qu'étant donné deux configurations qu'il ne peut distinguer, l'agent α

doit prendre la même décision dans les deux cas :

$$\pi_\alpha(x) = \pi_\alpha(y) \Rightarrow s_\alpha(x) = s_\alpha(y)$$

On peut de manière équivalente doter chaque agent α d'une relation d'équivalence $\sim_\alpha : \Omega \times \Omega \to \mathbb{B}$ indiquant que deux configurations sont *indistinguables* de son point de vue. Celle-ci peut être définie à partir de π_α par $x \sim_\alpha y \Leftrightarrow \pi_\alpha(x) = \pi_\alpha(y)$. Dans la mesure où une stratégie uniforme donne un même résultat pour deux configurations $x \sim_\alpha y$, cela revient à considérer une stratégie comportementale locale définie sur l'ensemble quotient $\mathfrak{X}_\alpha = \Omega_{/\sim_\alpha}$:

$$s_\alpha : \mathfrak{X}_\alpha \to \mathbb{A}_\alpha$$

En algorithmique répartie, la perception de l'environnement par un agent n'est plus une simple projection π_α fournie par le modèle, mais est plutôt "calculée" dynamiquement.

Topologie "épistémique". La représentation de la vision partielle d'un agent par une relation d'équivalence peut être généralisée [6]. Soit ϕ un ensemble de configuration (un *prédicat*). Le système peut être actuellement dans l'une des configurations de ϕ sans pour autant qu'un agent α le sache (il ne dispose pas de toute l'information). On définit $\kappa_\alpha(\phi)$ comme l'ensemble des configurations dans lesquelles α sait qu'il se trouve dans une configuration de ϕ. Cet opérateur de connaissance κ_α doit assez naturellement vérifier un certain nombre d'axiomes :

A0 $\kappa_\alpha(\Omega) = \Omega$ (hypothèse de monde clos : l'agent sait toujours qu'il se trouve dans une configuration de Ω) ;

A1 $\forall(\phi, \psi) \in \mathbb{P}(\Omega)^2,\ \kappa_\alpha(\phi \cap \psi) = \kappa_\alpha(\phi) \cap \kappa_\alpha(\psi)$ (axiome de distribution) ;

A2 $\forall \phi \in \mathbb{P}(\Omega),\ \kappa_\alpha(\phi) \subseteq \phi$ (axiome de vérité : l'agent ne peut connaître que des vérités[6]) ;

A3 $\forall \phi \in \mathbb{P}(\Omega),\ \kappa_\alpha(\phi) \subseteq \kappa_\alpha(\kappa_\alpha(\phi))$ (axiome d'introspection positive) ;

Cela revient à définir κ_α comme un opérateur d'intérieur (ouverture topologique) sur l'espace Ω : *contractant* [A2], *idempotent* [A2+A3], et *"stable"* (par intersection finie) [A0+A1]. L'ensemble image $\mathcal{T}_{\kappa_\alpha} = \mathrm{img}(\kappa_\alpha) = \{\kappa_\alpha(\phi) \mid \phi \in \mathbb{P}(\Omega)\}$ définie une topologie sur Ω. En fait, étant donné un ensemble de configurations ϕ, l'opérateur κ_α associe l'intérieur de ϕ pour la topologie $\mathcal{T}_{\kappa_\alpha}$ (le plus grand ouvert de $\mathcal{T}_{\kappa_\alpha}$ inclus dans ϕ). Ainsi, un agent ne connaît pas exactement la configuration actuelle du système mais il sait qu'il est dans son adhérence. Cela généralise l'approche par les relations d'indistinguabilité où l'adhérence d'une configuration est donnée par sa classe d'équivalence pour la relation $\sim_\alpha$ (correspondant au concept d'*information set*). En ajoutant l'axiome [A4], ci-dessous, on exprime le fait que tout ouvert est également un fermé ce qui fait de $\mathcal{T}_{\kappa_\alpha}$ une topologie totalement discontinue. On peut alors montrer qu'une telle topologie découle nécessairement d'une relation d'équivalence $\sim_\alpha$ et qu'inversement une topologie associée à une relation d'équivalence vérifie cet axiome.

A4 $\forall \phi \in \mathbb{P}(\Omega),\ \complement\kappa_\alpha(\phi) \subseteq \kappa_\alpha(\complement\kappa_\alpha(\phi))$ (axiome d'introspection négative).

La notion généralisant de manière naturelle le concept de stratégie uniforme est celle de *stratégie continue* $s_\alpha : \langle \Omega, \mathcal{T}_{\kappa_\alpha} \rangle \to \langle \mathbb{A}_\alpha, \mathbb{P}(\mathbb{A}_\alpha) \rangle$. Ainsi dans le cas d'une topologie discrète $\mathbb{P}(\mathbb{A}_\alpha)$ sur l'ensemble des actions de α, pour toute action que l'agent entreprend dans une configuration ω, il

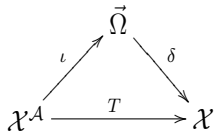
[6] C'est en particulier ce qui distingue une connaissance d'une croyance.

existe un ouvert O de configurations autour de ω tel que l'agent doive y prendre la même décision.

Exécution et protocole. Une *exécution* de longueur k (un chemin de longueur k) sur $\langle \Omega, \mathbb{A}, \xrightarrow{\bullet} \rangle$ est une suite de configurations $(\omega_0, \cdots \omega_k)$ en alternance avec des événements $(e_1, \cdots, e_k)$ telle que pour tout $i \in [\![1, k]\!]$, on ait $\omega_{i-1} \xrightarrow{e_i} \omega_i$:

$$\omega_0 \xrightarrow{e_1} \omega_1 \xrightarrow{e_1} \omega_2 \cdots \omega_{k-1} \xrightarrow{e_k} \omega_k$$

On note $\vec{\Omega}^k$ l'ensemble de ces chemins de longueur k et $\vec{\Omega} = \bigoplus_{k \in \mathbb{N}} \vec{\Omega}^k$ l'ensemble de toutes les exécutions finies. L'ensemble $\vec{\Omega}^0$ est identifié à Ω ce qui permet de plonger ce dernier dans $\vec{\Omega}$. La construction faite précédemment sur l'ensemble des configurations peut alors facilement être généralisée à l'ensemble des exécutions $\vec{\Omega}$. En particulier, une notion de topologie peut être définie sur l'ensemble des exécutions et lorsque l'environnement est déterministe (mémoire partagée), l'algorithme donné en figure 2 permet de ramener l'étude de cet espace d'exécution à l'étude de l'espace des ordonnancements Σ. L'idée est alors de représenter un protocole de décision comme une fonction continue $\delta : \vec{\Omega} \to \mathcal{X}$ (ou $\delta : \Sigma \to \mathcal{X}$) à valeur dans l'ensemble $\mathcal{X}$ des décisions [9] :



3 Système de processus asynchrone

Trois problématiques sont relativement classiques en informatique répartie, à savoir les *pannes de l'environnement* [5], les *processus byzantins* (agents jouant contre le système [7]) et l'*asynchronisme* [3]. Nous avons en particulier cherché à comprendre ce qu'impliquent les hypothèses faites sur le système dans chacun de ces cas en termes topologiques. Par manque de place, nous ne présenterons ici que les résultats concernant le troisième point, à savoir les résultats sur les systèmes asynchrones. Dans ce cadre, le résultat suivant est classique :

FISCHER, LYNCH & PATERSON, 1983 : *Dans un système totalement asynchrone, composé de processus déterministes, le problème du consensus est insoluble à partir du moment où un seul processus est incorrect [3].*

Si n désigne le nombre d'agents, on dit que deux configurations sont adjacentes si $(n-1)$ agents ne peuvent les distinguer :

$$\sim = \bigcup_{|S| \geq n-1} \bigcap_{\alpha \in S} \sim_\alpha$$

L'idée de la preuve de ce théorème est alors simple : on commence par montrer que l'ensemble des configurations initiales $\mathcal{X}^{\mathcal{A}}$ est connecté pour la relation $\sim$. Puis, on montre que pour toute configuration ω, l'ensemble de ses successeurs immédiats est également connecté pour la relation $\sim$. On en déduit de proche en proche une connexité au niveau des exécutions $\vec{E} = \bigcup_{x \in \mathcal{X}^{\mathcal{A}}} \vec{\Omega}(x, \bullet)$. Le théorème FLP se ramène ainsi à une propriété de connexité sur l'ensemble des exécutions. On peut en effet partitionner l'ensemble des exécutions $\vec{E}$ en F_0 (*resp.* F_1) : celles pour lesquelles certains agents décident 0 (*resp.* 1). Imposer la non trivialité et la terminaison du protocole revient à dire que l'ensemble des exécutions intersecte simultanément F_0 et F_1 et qu'il est inclus dans leur union. D'après la connexité de $\vec{E}$, on en déduit que $F_0 \cap F_1 \neq \emptyset$ et donc qu'il existe des exécutions pour lesquelles

certains agents décident 0 et d'autres 1. D'où l'impossibilité d'obtenir systématiquement un consensus.

Connexité de plus haut niveau. On se rend ainsi compte que le problème posé par le consensus est finalement assez simple puisqu'il se ramène à une simple notion de connexité. Des problèmes plus généraux tels que l'accord dans un k-ensemble [9] (au plus k valeurs distinctes peuvent être choisies par les agents) n'ont pas de solutions aussi évidentes. C'est dans ces cas là que les résultats de topologie montrent tout leur intérêt. On ne fait alors plus appel à une notion de connexité (0-connexité $\approx$ il est toujours possible de relier deux points par un chemin continu), mais à des notions de connexité d'ordre supérieur (k-connexité $\approx$ il est toujours possible d'étendre continument une k-sphère en une $(k+1)$-boule).

Poids d'un agent. Au début de l'article, nous rappelions la notion de poids décisionnel d'un agent. Cependant le poids de l'agent dans la décision dépend de son poids dans le calcul, c'est à dire du temps qui lui est accordé par l'ordonnanceur. On a ainsi affaire à deux notions de pondération. En décision classique, le poids d'un agent dans la décision est donné par un jeu coalitionnel (ou une fédération). La prise en compte de la dynamique superpose un autre poids relatif au temps de calcul imparti à un agent par l'ordonnanceur.

On ne peut donc pas a priori *garantir d'accorder un poids décisionnel précis à un agent car l'ordonnanceur peut moduler ce poids en terme de temps de calcul : un agent, aussi influent soit-il, qui ne dispose d'aucun temps de calcul est nécessairement de poids nul dans la décision.*

4 Conclusion

Pour conclure, nous synthétisons quelques réflexions inspirées par notre étude et qui constituent, à notre avis, des pistes de recherche à approfondir.

Mélange des problèmes. Tout d'abord, il est difficile de prendre en compte l'ensemble des problèmes : environnement incorrect, agents byzantins et ordonnanceur. En fait, les problèmes d'asynchronisme devraient théoriquement empêcher la coordination des byzantins autant que des agents corrects. On suppose ainsi que l'ordonnanceur, l'environnement et les byzantins agissent de concert contre les agents corrects. De leur côté, les agents corrects forment eux-mêmes une coalition pour jouer contre le système et perdent du même coup leurs propres intérêts. On se ramène donc à un jeu à deux coalitions et à somme nulle permettant de faire une étude dans le pire des cas : l'ensemble des agents corrects unis contre le reste du système.

Connaissance *v.s.* rationalité. Une décision est simultanément guidée par la connaissance qu'ont les agents de la situation et par leurs préférences. Cependant en théorie des jeux, la synchronisation est acquise et la communication peu fréquente. Le modèle des connaissances est basique et la notion de préférence relativement fine. En informatique répartie, le modèle des connaissances est plus subtile, il prend en compte la difficulté posée par la synchronisation, mais la notion de préférence est grossière (les buts acceptables sont les configurations cohérentes, sinon ils sont inacceptables). De plus, tous les agents corrects sont d'accord sur le but (situation non-compétitive).

SMA et informatique répartie. L'informatique répartie se donne souvent pour but de *dessiner les contours de ce qui est réalisable*. Elle étudie ainsi des conditions extrêmes. Le domaine des systèmes multia-

gents se veut plus pragmatique en se plaçant dans des *contextes plus consensuels*. Il faut donc faire des compromis parfois difficiles entre des critères "naturels" tels que la sûreté ou la vivacité et d'autres critères dépendant de l'application (rationalité encapsulée au niveau des agents). Ce compromis cohérence/vivacité/rationalité reste à établir : l'informatique répartie se concentre sur les deux premiers points tandis que les SMA (et la théorie des jeux) développent la notion de rationalité locale aux agents, mais insistent moins sur la cohérence/vivacité.

Références

[1] J.-P. Barthélemy and M.F. Janowitz. A formal theory of consensus. In *Siam. J. Discr. Math.*, volume 4, pages 305–322, 1991.

[2] C. Chichilnisky and G. Heal. Necessary and sufficient conditions for a resolution of the social choice paradox. *Journal of Economic Theory*, 31 :68–87, 1983.

[3] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2) :374–382, 1985.

[4] J. Goubault-Larrecq. Une introduction aux capacités, aux jeux et aux prévisions. Technical report, INRIA Futurs projet SECSI, mars 2006.

[5] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3) :549–587, 1990.

[6] F. Koessler. Common knowledge and interactive behaviors : A survey. *European Journal of Economic and Social Systems*, 14(3) :271–308, 2000.

[7] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3) :382–401, 1982.

[8] N.A. Lynch. *Distributed Algorithms*. Morgan-Kaufmann, 1996.

[9] M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible : the topology of public knowledge. In *STOC'93*, pages 101–110. ACM Press, 1993.